Major project report on

"COVID-19 OUTBREAK IN INDIA – EXPLORATORY DATA ANALYSIS AND PREDICTION"

By:

ANUJ KUMAR (2K16/SE/015)
CHANDRABHAN (2K16/SE/023)
DEEPANSHU SINGH (2K16/SE/024)

Under the guidance of:

Dr. RAJNI JINDAL

Professor & HOD, Dept. of Computer Science & Engineering
Delhi Technological University

Submitted in fulfillment of the
Degree of Bachelor of Technology
In

Software Engineering
(Batch: 2016-2020)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY (FORMERLY DCE)

DELHI-110042 (INDIA)

DECLARATION

We declare that the work embodied in this project report titled "**COVID-19 in India – Exploratory Data Analysis and Prediction**", forms our own contribution of work under the guidance of **Dr. Rajni Jindal** at the Department of Software Engineering, Delhi Technological University. All information included from any other source has been duly acknowledged.

Anuj Kumar (2K16/SE/015)
Chandrabhan (2K16/SE/023)
Deepanshu Singh (2K16/SE/024)

CERTIFICATE

This is to certify that **Anuj Kumar**, **Chandrabhan** and **Deepanshu Singh**, students of B.Tech Software Engineering, Delhi Technological University have successfully completed the project titled “COVID-19 in India – Exploratory Data Analysis and Prediction” under the guidance of **Dr.Rajni Jindal**.

Dr. Rajni Jindal
Professor
Dept. of Computer Science and Engineering
Delhi Technological University

Anuj Kumar (2K16/SE/015)
Chandrabhan (2K16/SE/023)
Deepanshu Singh (2K16/SE/024)

ACKNOWLEDGMENT

No project has ever been created without the assistance and guidance of other people. This one is certainly no exception. We take this opportunity to show gratitude towards the people who helped us in the making of this project. We are extremely thankful and would like to express deep regard towards the faculty for encouraging us, clearing our doubts and constantly supporting us throughout this project. We are indebted to **Dr. Rajni Jindal** who has been a constant support and her exceptional mentoring. Her valuable suggestions and ideas always proved helpful for us and bridged the gap in our knowledge of the subject.

ABSTRACT

As of April 30th, 2020, the total number of confirmed cases and deaths for the COVID-19 (Coronavirus) outbreak in India was 33050 and 1074. It is a disturbing scenario, as India will enter stage-3, termed as the community level transmission, of COVID-19, with such a massive population within a few days. Because of the non-existence of an appropriate cure or vaccination, and with an imperfect grasp of the existing epidemiological situation, quantitative mathematical models can aid investigating both propagation and regulation of COVID-19. In this study, we consider mathematical models on COVID-19 transmission that combines lock-down effect and transmission variation between populations. We have strived to explore the COVID-19 data to visualize the spread of the virus in the country. We dissected the data to extract the state-wise insights which can reveal a lot of information about the worse condition of a region compared to any other and therein we can find the answer of the question, what can be done to improve the situation in worst effected states like Maharashtra, Gujarat and Delhi? After that we will do a comparison of the Indian situation with other countries to get an idea of the stage of the pandemic at which we are in. Finally, we used state of the art forecasting models to simulate and predict the number of confirmed cases and deaths in the next 15 days. Using daily reported cases of COVID-19 from India, we analyze the impact of the two lock-downs in terms of reduction in the number of cases and deaths. There's a ray of hope for India as our forecast suggests that lock-down would put a big dent in the rising number of deaths and confirmed cases. Further lock-down extension in the country could put India in a comfort zone. Finally, we suggest a policy for the Indian government to control COVID-19 outbreak. There is an obvious benefit to building a data-driven exploration and prediction support tool for both government and medical professionals to help them combat this pandemic. These tools use the data gathered after the pandemic started and made its way out of Mainland China where all of this started. With the advent of modern machine learning algorithms it is now possible to get a much more sophisticated extrapolation and information from the available data.

Keywords: COVID-19, Infectious Disease, Machine Learning models, Lockdown, Forecasting, Pandemic, Growth Factor.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

WHO:	World Health Organization
RNA :	Ribonucleic Acid
COVID:	Corona Virus Disease
SARS-CoV:	Severe acute respiratory syndrome Coronavirus
MERS-CoV:	Middle East respiratory syndrome-related Coronavirus
ARIMA:	Auto Regressive Integrated Moving Average
XGB :	Extreme Gradient Boosting
LGBM:	Light Gradient Boosting Model
RMSE:	Root Mean Square Error
MAE:	Mean Absolute Error
ICMR:	Indian Council of Medical Research
UT:	Union Territory
DNN:	Deep Neural Network
LSTM:	Long Short Term Model

Chapter 1

Introduction

1.1 Background

The Coronavirus pandemic, as declared by WHO, started in the Chinese city of Wuhan on 31th December,2019. The virus has spread rapidly and killed over 238,198 people worldwide. Till April 30th, 2020, over 3,096,626 infections—spanning 210 countries and territories as reported by the World Health Organization (WHO) [1]. The WHO has already declared outbreak as a pandemic. Coronaviruses are wrapped non-divided positive-sense RNA infections that belong to the Coronaviridae family and the request Nidovirales, and are broadly disseminated among people and different warm blooded animals [2]. The coronavirus, COVID-19 began in territory of China, with a land accentuation at Wuhan, the capital city of Hubei region and has generally spread everywhere throughout the world. Huge numbers of the underlying cases were typically acquainted with the wholesale fish market of Hunan, which additionally listed live creatures. Clinical preliminaries of hospitalized patients found that patients display side effects predictable with viral pneumonia at the beginning of COVID-19, most regularly fever, hack, pharyngitis and exhaustion .A few patients detailed changes in their ground-glass lungs; typical or lower than normal white lymphocyte platelet tallies and platelet checks; hypoxemia; and unsettled liver and kidney work [2]. Most were said to be topographically identified with the wholesale fish market of Hunan. Extreme episodes happen in USA (3,67,004 cases), Italy (1,32,547 cases), Spain (1,36,675 cases), Germany (103,375 cases), France(98,010), China (81,708 cases) thus numerous nations and the illness keeps on spreading comprehensively [3]. This has been proclaimed a pandemic by the WHO (World Health Organisation). It is the third zoonotic human coronavirus that has emerged in the current century, after the 2002 serious intense respiratory condition coronavirus (SARS-CoV), which spread to 37 nations and the 2012 Middle East respiratory disorder coronavirus (MERS-CoV), which spread to 27 nations [2]. The 2019 pandemic novel coronavirus was first affirmed in Quite a while on 30 January 2020, in the province of Kerala [4]. A sum of 4778 acclimated cases, 382 recuperations and 136 passings in the nation have been accounted for starting at 6 April 2020 [5]. The Indian government has presented social separation as a precautionary measure to keep away from the chance of an enormous scope populace development that can quicken the spread of the malady. India government executed a 14-hour deliberate open time limit on 22 March 2020. Besides, the Prime Minister of India likewise requested an across the country 21-day lockdown at 12 PM on 24 March to slow the spread of COVID-19, influencing India's whole 1.3 billion population.

Notwithstanding no immunization, social separating has distinguished as the most usually utilized anticipation and control methodology. The reason for these activities is the

limitation of social connection in work environments, schools, and other open circles, aside from basic open administrations, for example, fire, police, medical clinics. Presumably the spread of this infection episode has genuinely disturbed the life, economy and wellbeing of residents [6]. This is an incredible worry for everybody to what extent this situation will last and when the sickness will be controlled. Numerical displaying dependent on arrangement of differential conditions may give a far reaching component for the elements of COVID-19 transmission. A few displaying contemplates have as of now, been performed for the COVID-19 episode. They additionally determined around 2.68 is the essential conceptive number for COVID-19 [7]. Scientists found that the measure of control generation number might be as high as 6.47, and that techniques for mediation including serious touch followed by isolation and detachment would adequately limit COVID cases [7]. For the fundamental regenerative number, scientists revealed an estimation of 3.1 dependent on the information fitting of a SEIR model, utilizing a supposition of Poisson-appropriated day by day time increases [8,23]. A report by Cambridge College has shown that India's countrywide three-week lockdown would not be satisfactory to forestall a resurgence of the new coronavirus pestilence that could skip back in months and cause a large number of diseases. They proposed that a few lockdowns can expand the stoppage longer with five-sunrises in the middle of or a solitary 49-day lockdown. Information driven scientific displaying assumes a key job in ailment counteraction, making arrangements for future flare-ups and deciding the viability of control. A few information driven displaying tests have been acted in different locales.

1.2 Motivation

COVID-19 has had a near-unprecedented impact on the modern world, this pandemic has shown our inability of dealing with catastrophes in such a definite way. This has been the first situation after world war 2, that has put the world economy to a halt for such a long period of time and has affected the livelihoods of average people all over the world. No other response to a crisis led to the home confinement of more than half of the world population. The sheer amount of deaths caused by this pandemic and even deaths of those fighting on the front lines to eradicate this virus is astounding. The massive psychological damage that has been done to the people confined in their homes is worrysome. The massive amount of working class population and daily labourers that have lost their jobs and are away from their homes and cannot get back because of the restrictions is a tragedy in and of itself. On the other note, the remarkable works done by many studies over the years in the field of Infectious disease modelling has also played a role in the making of this paper. We hope this research project can help create some valuable insights which will accelerate our fight against this pandemic in India.

1.3 Objective

Our objective is to analyse the data to understand the way in which the virus spread across India, identify the patterns of the outbreak and forecast trends in the spread of cases in India (and various states of India) using Machine Learning techniques. We aim to predict the number of cases that are going to come up in next 10-15 days. Also at the present moment, there are extremely less studies that contemplate the effect of lock-down on COVID-19 transmission elements in India. Therefore, we also intend to analyze the lockdown periods imposed by the government of India and find out if it is an effective strategy in this pandemic and suggest if it should be extended or not. It should be noted that this study was performed on the data from 30th January 2020 (the first reported case in India) till 30th of April 2020.

Chapter 2

Related Work

In the field of epidemiology and infectious disease modelling, many different types of models have been proposed by various researches in the past years. Techniques like SIR and SEIR models of infectious diseases have been used to understand and predict the spread of a viral disease through a population. These techniques divide a population in different compartments such as Susceptible, Infected, Exposed, and Recovered and then combine some parameters with these groups to form a mathematical model. Studies like "Predictions for COVID-19 Outbreak in India using epidemiological models" by "Rajesh Ranjan" [22] use the above described model for simulating the epidemic to find the reproduction number (R_0) of the coronavirus. Another study "Propagation Analysis and Prediction of the COVID-19" [18] came out of China which used Gaussian distribution methods to model the outbreak using the data from the Hubei province in China and produced very good results. They predicted the latency period and basic reproduction number of the virus. They also did the comparison of the outbreak of the virus in Hubei province in China with other countries currently battling with the virus which is what we have tried to do in our case as well.

By taking inspiration from above studies and at the same time using our own methods to analyze the pandemic in the case of India, we have aimed to forecast the trend in increase in number of cases in the country. Moreover, we have also done state-wise analysis to understand the effect of the virus locally. We also compare the current stage of India with other countries that are ahead in the outbreak than India. This can help us in taking measures that can help save thousands of lives. As discussed in the previous section, we also use growth factor modelling technique to understand the effect of lockdowns in combating the virus.

It is very important to use modern state of the art forecasting techniques to simulate the spread of an outbreak as dangerous and resilient as the COVID-19. Using these models, we can analyze large amounts of data with minimal efforts and extract very detailed information out of the datasets. This can help in effective decision making and utilizing our resources in the most optimal way possible.

Chapter 3
Research Methodology

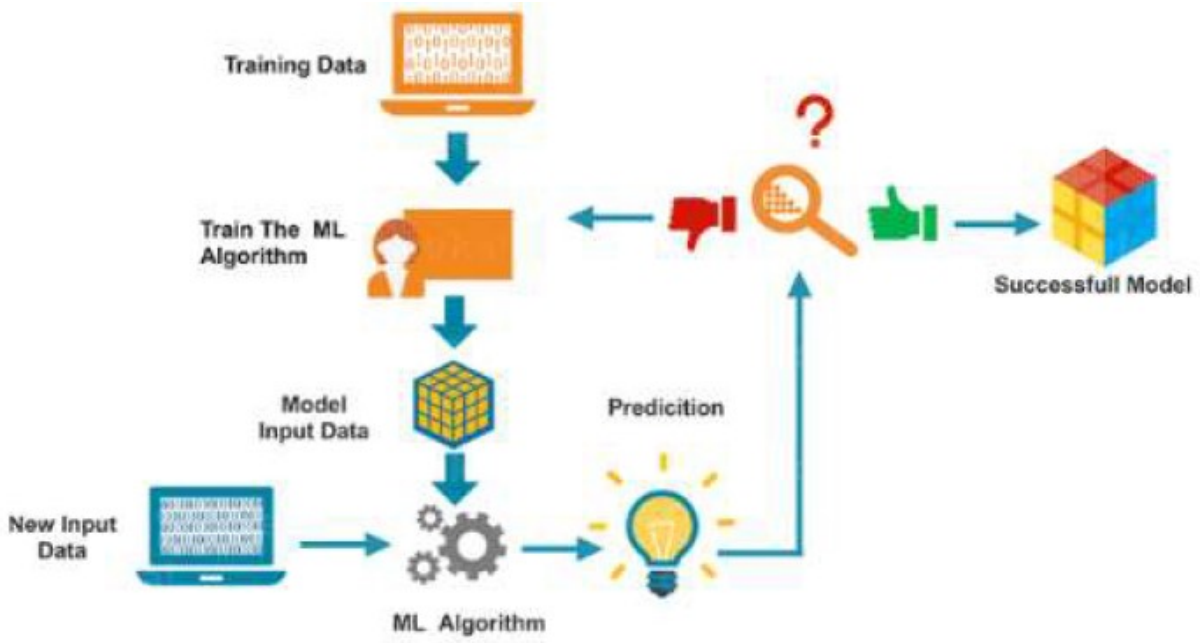

Fig. 1. Overview of the machine learning methodology

3.1 Software Requirements

This project uses the following software, tools and Python libraries.

- Jupyter Notebook
- Anaconda
- Python 3
- NumPy
- Pandas
- Scikit-learn
- Matplotlib
- SciPy

3.2 Overview of the process

The datasets we used for the Indian dataset (refer section 4.1), we grouped the data by dates to get the numbers of all the daily cured, recovered and deceased cases from 30-01-2020 to 30-04-2020. We used this to visualize the trend in increase in the number of cases (see figure 9). We also extracted some state-wise insights from the data by grouping the dataset according to the state or union territories (see figure 12). The international data from John Hopkins University (refer section 4.1) is used to compare the stage of transmission of the virus within India to other countries of the world that are ahead of us in the pandemic (see figure 18). According to the daily increase in the number of cases, the growth factor is calculated and using that we forecast the number of cases for the next 15 days i.e., from 01-05-2020 to 15-05-2020. The data was organized to be suitable as per the input requirements of the forecasting models used (refer section 3.3) and the number of cases were forecasted using these models for the duration of 15 days. The traditional machine learning algorithms like, random forest and gradient boosting frameworks are used and evaluated according their error rates and optimal algorithms are suggested. The two lock-downs in India, are studied in three stages (a) the period from the first recorded case (30-01-2020) to the start of first lockdown (25-03-2020). This period could be used as a benchmark to analyze the effectiveness of lockdown, (b) Lockdown 1.0, started from 25-03-2020 and lasted 21 days and (c) Lockdown 2.0, started from 14-04-2020 and lasted 19 days [20]. In each case the growth factor from the number of cases are calculated and are compared with the forecasted models used by us. The growth factor is calculated by taking the average of differences of the number of cases on each day with the previous day. This gives us a good idea of the effectiveness of the measures taken by the government and whether or not it should be continued.

3.3 Model description

3.3.1 Prophet Forecasting Model

Prophet is a strategy for gauging time arrangement information dependent on an added substance model where nonlinear patterns are fit with yearly, week by week, and day by day regularity, in addition to occasional impacts. It works best with time arrangement that have solid occasional impacts and a few periods of verifiable information. Prophet is hearty to missing information and moves in the pattern, and ordinarily handles exceptions well [9].

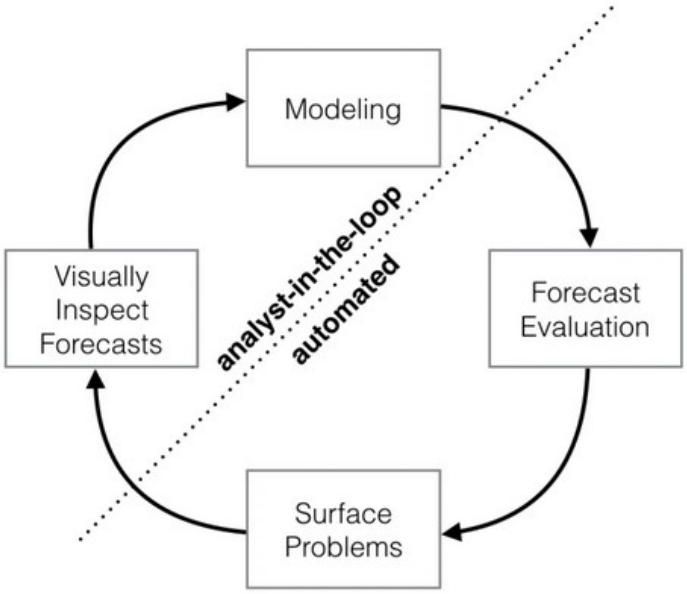

Fig. 2. How prophet model functions

Advantages of using prophet model :

Exact and quick : Prophet is utilized in numerous applications across Facebook for delivering solid conjectures for arranging and objective setting.

Tunable conjectures : The Prophet technique incorporates numerous opportunities for clients to change and modify figures. You can utilize human-interpretable parameters to improve your conjecture by including your area information.

Completely programmed : Get a sensible estimate on muddled information with no manual exertion. Prophet is vigorous to anomalies, missing information, and emotional changes in your time arrangement.

Working of the Model:

At its center, the Prophet strategy is an added substance relapse model with four primary segments:

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

g(t) models pattern, which depicts long haul increment or decline in the information. Prophet joins two pattern models, a soaking development model and a piecewise straight model, contingent upon the kind of determining issue,

s(t) models regularity with Fourier arrangement, which depicts how information is influenced via occasional factors, for example, the season (for example more scans for eggnog throughout the winter occasions),
h(t) models the impacts of occasions or huge occasions that sway business time arrangement (for example new item dispatch, Black Friday, Superbowl, and so forth.),
ϵ_t speaks to a final blunder term [10].

Prophet model likewise considers the regularity part of the pandemic but since of the brief time of the episode that usefulness isn't utilized here for anticipating the spread of the infection. Figure 3 shows the vulnerability levels of forecast is low after utilizing this model.

3.3.2 ARIMA Model

ARIMA represents **A**uto **R**egressive **I**ntegrated **M**oving **A**verage.

AR (AutoRegression): A model that utilizes the reliant connection between a perception and some number of slacked perceptions. p is a parameter of what number of slacked perceptions to be taken in.

$$X_t = c + \sum_{i=1}^{p} \varphi_i X_{t-i} + \varepsilon_t$$

I (Integrated): A model that utilizes the differencing of crude perceptions (for example taking away an perception from the past time step). Differencing in measurements is a change applied to time-arrangement information so as to make it fixed.

$$\begin{aligned} y_t^* &= y_t' - y_{t-1}' \\ &= (y_t - y_{t-1}) - (y_{t-1} - y_{t-2}) \\ &= y_t - 2y_{t-1} + y_{t-2} \end{aligned}$$

MA (Moving Average): A model that utilizes the reliance between a perception and a lingering mistake from a moving normal model applied to slacked perceptions. q is a parameter of what number of slacked perceptions to be taken in. In spite of the AR model, the limited MA model is consistently fixed.

$$X_t = \mu + \varepsilon_t + \theta_1 \varepsilon_{t-1} + \cdots + \theta_q \varepsilon_{t-q}$$

Assumptions:

ARIMA model depends on various presumptions, for example, Data doesn't contain irregularities, Model parameters and mistake term is consistent, historic time points direct conduct of present time points which probably won't hold in focused on advertise information conditions, Time arrangement is fixed [11].

Parameters of the ARIMA model:

- **p** (slack request) is the quantity of slack perceptions remembered for the model,
- **d** (level of differencing) is the occasions that the crude perceptions are differenced,
- **q** (request of moving average)is the size of the moving normal window.

3.3.3 Random Forest Regressor

The Random Forest is an algorithm equipped for performing both joining and categorizing undertakings with the use of more than one decision trees and a procedure called “Bootstrap Aggregation", also called “bagging” [12]. This gives more reliable result because now we do not depend on a single decision tree. We can tune various hyper-parameters to get better output. The random forest model also avoids over-fitting because of grouping characteristic of the model by a subset of features.

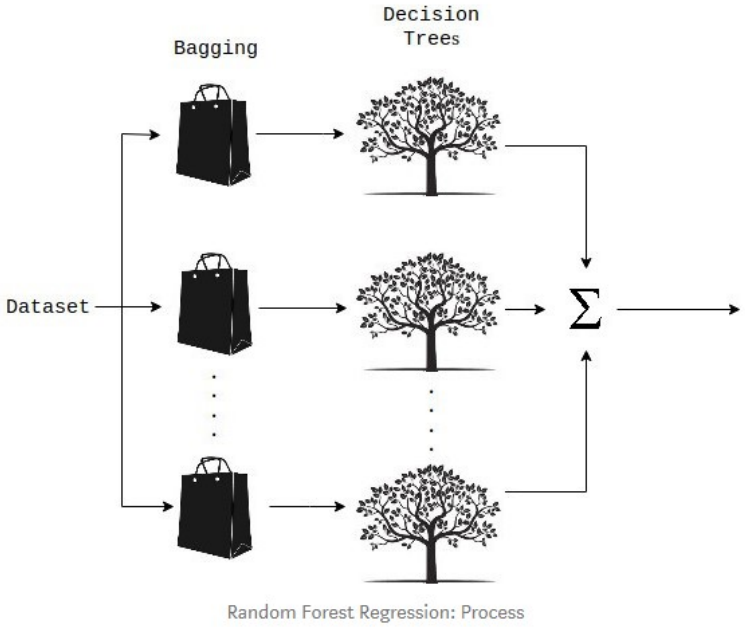

Fig. 3. How random forest processes input data to get relevant results

Random Forest divides the dataset in different compartments and based on the parameters passed as input to the algorithm it evaluates each grouping and then combines all the outputs to give a cumulative result. This method is very effective and avoids over-fitting.

3.3.4 Gradient Boosting Frameworks

Gradient Boosting frameworks greatly increase the functionality of normal decision trees. It produces a model based on a lot of weak models and the results are generalized and combined using a differentiable loss function [13]. Here, we will use two gradient boosting frameworks, namely, **XGBoost** and **Light GBM**.

3.3.4.1 Light GBM

Light Gradient Boosting is a modified tree based algorithm. It boosts performance by using loss functions to evaluate which branches of the tree will give better results and explores them leaves out the rest.

Below diagrams explain the implementation of Light GBM and its comparison with other boosting algorithms.

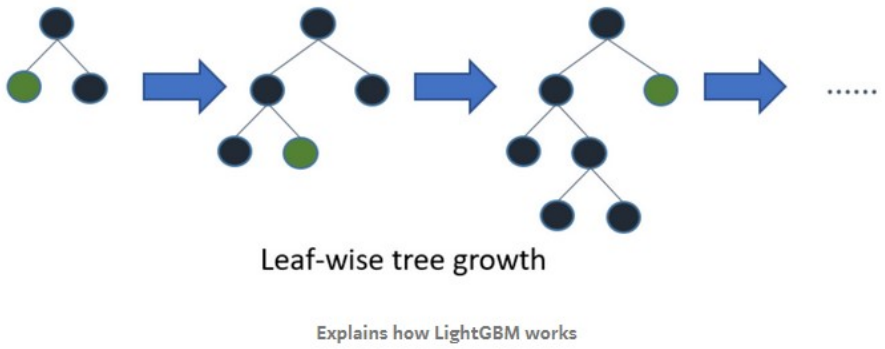

Fig. 4. Working of the Light Gradient Boosting Model

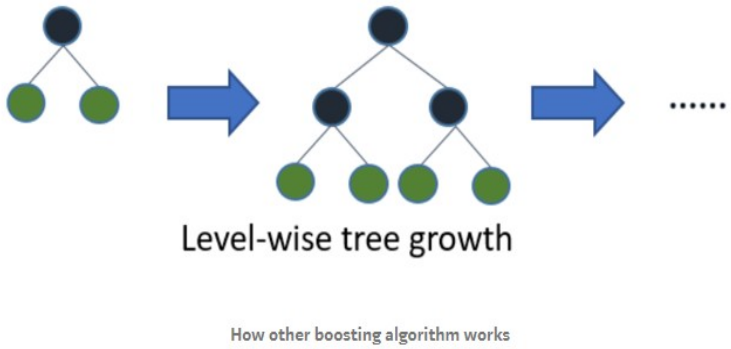

Fig. 5. How other Gradient Boosting algorithms work

Light gradient boosting has many plus points such as, its ability for faster training, high accuracy and efficiency and its ability to handle large datasets. We utilized this algorithm for our prediction.

3.3.4.2 XGBoost

XGBoost represents the Extreme Gradient Boosting. It has perhaps quickest execution using decision trees. It is a choice tree-based group Machine Learning calculation that utilizes an angle boosting structure [14].

Below figure shows the evolution of simple decision trees to the Gradient Boosting models like XGBoost.

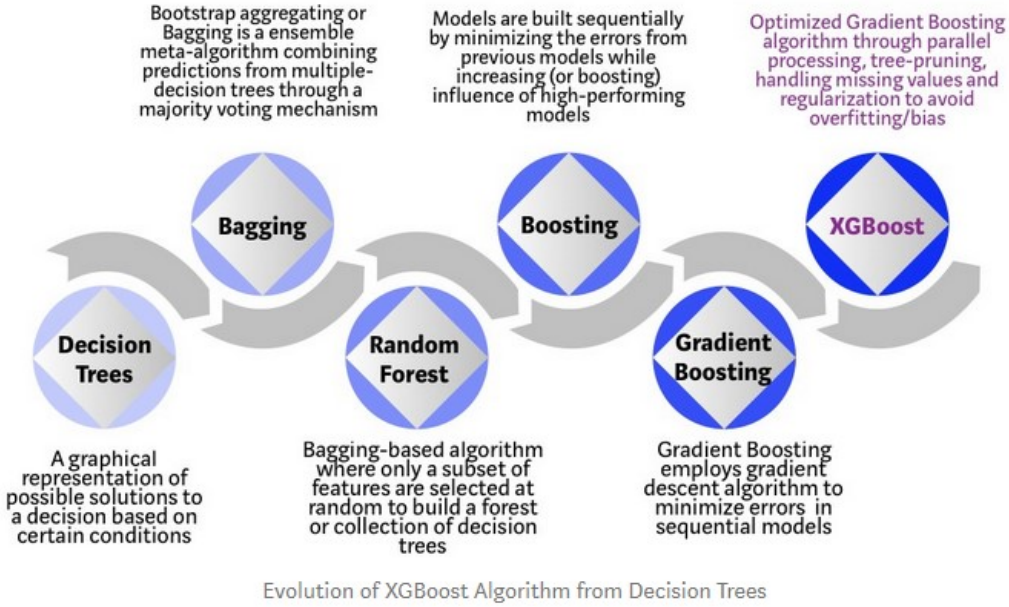

Fig. 6. Evolution of Gradient Boosting Frameworks from Decision trees to XGBoost algorithm.

In prediction scenarios including unstructured information (pictures, content, and so on.) ANN systems will in general beat every single other calculation or structures. Be that as it may, with regards to little to-medium organized/plain information, choice tree based calculations are viewed as best-in-class at the present time.

3.4 Performance Measures

For the scope of this study, we are going to use the Root Mean Square error (RMSE) metric to evaluate the performance of our models on the dataset. Root Mean Square error is a quadratic method that measures the average error in a set of predictions. It is calculated by taking the square root of the average of the squares of differences between predicted values and actual values.

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{j=1}^{n}(y_j - \hat{y}_j)^2}$$

Since RMSE is quadratic in nature, it takes into account the larger error values more than the lower error values which can be advantageous in many cases. RMSE values is better than other metrics like Mean Absolute Error (MAE) because it avoids taking absolute values which is not desirable in mathematical calculations.

We use the RMSE scores to evaluate the machine learning models used to display the errors in the prediction of the number of confirmed COVID-19 cases in the country on the test set that we divided beforehand, after training the models on the training set under optimal parameters.

Chapter 4

Experimental Design

4.1 About Data

- The time-series data is collected from the Indian Health Ministry Website (https://www.mohfw.gov.in/). Each row contains data for each state with at least one confirmed case. There are different columns in the dataset such as Date, State, Latitude, Longitude , number of Confirmed, Cured and Death cases. There are 1464 instances in the dataset spanning from 30-01-2020 to 30-04-2020.

- For comparison with other countries, we have used the time-series data from the website https://github.com/datasets/covid-19 which is maintained by Johns Hopkins University Center for Systems Science and Engineering (CSSE).

4.2 Data Exploration

In this section, we explore the data to gather insights about the change in the number of cases using different features or variables. We will use python's modules like pandas, matplotlib and seaborn to visualize and understand the dataset. First, we will gather the data from above sources into panda's dataframes so that it is easier for us to visualize using python. After that, we analyze the spread of cases among age and gender groups in the country. We also do the ICMR test (test for COVID-19) analysis. We also analyze the ICMR testing labs across the country. State-wise analysis of ICMR tests is also done and their results are analyzed. Healthcare analysis is also done with respect to parameters like number of available beds, healthcare facilities and hospitals in various provinces of the country. Finally, we use the international dataset to compare the stage in which India in comparison to other countries. Now, we will explore these datasets now in a detailed manner to extract useful insights on how to process and perform transformations on this data to make prediction for future. We also have the COVID-19 data from other countries as well to compare our conditions with them to get an idea of how well are we doing and also get some information about best practices that need to be followed to overcome this pandemic.

4.2.1 Gathering Data

We have our dataset divided into two parts :-

- **Indian Data**

```
age_details = pd.read_csv('../MajorProject/DataIndia/AgeGroupDetails.csv')
india_covid_19 = pd.read_csv('../MajorProject/DataIndia/covid_19_india.csv')
hospital_beds = pd.read_csv('../MajorProject/DataIndia/HospitalBedsIndia.csv')
individual_details = pd.read_csv('../MajorProject/DataIndia/IndividualDetails.csv')
ICMR_details = pd.read_csv('../MajorProject/DataIndia/ICMRTestingDetails.csv')
ICMR_labs = pd.read_csv('../MajorProject/DataIndia/ICMRTestingLabs.csv')
state_testing = pd.read_csv('../MajorProject/DataIndia/StatewiseTestingDetails.csv')
population = pd.read_csv('../MajorProject/DataIndia/population_india_census2011.csv')
```

- **Other Countries (For Comparison)**

```
world_population = pd.read_csv('../MajorProject/DataIndia/population_by_country_2020.csv')
confirmed_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/
deaths_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/css
recovered_df = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/
latest_data = pd.read_csv('https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/c
```

4.2.2 Data Analysis and Variable Description

- Understanding Indian Data

Age Group Disease Spread Analysis

```
labels = list(age_details['AgeGroup'])
sizes = list(age_details['TotalCases'])

explode = []

for i in labels:
    explode.append(0.05)

plt.figure(figsize= (12,7))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=9, explode =explode)
centre_circle = plt.Circle((0,0),0.70,fc='white')

fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('India - Age Group wise Distribution',fontsize = 20)
plt.axis('equal')
plt.tight_layout()
```

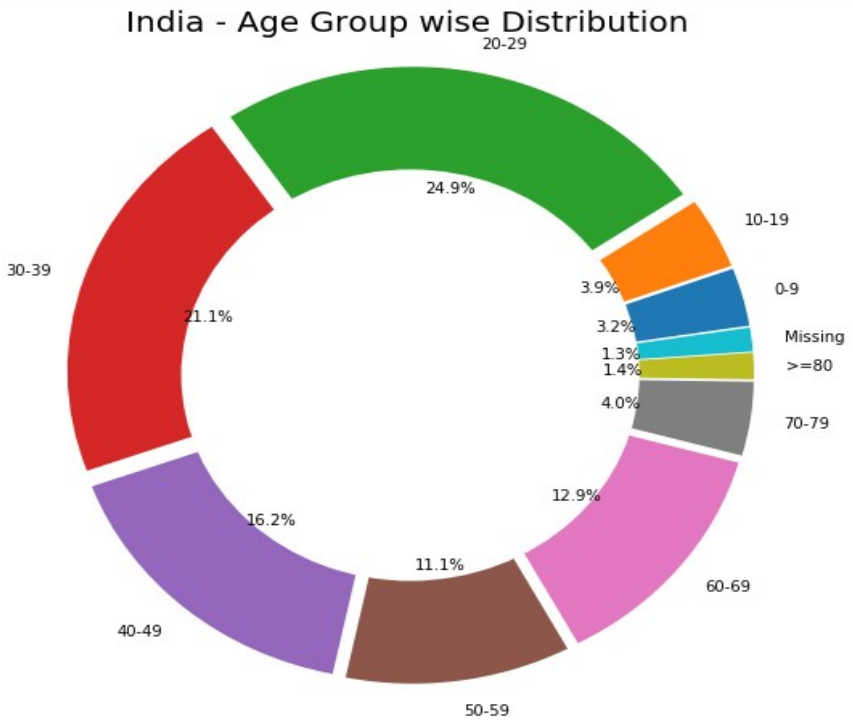

Fig. 7. Age-wise distribution of the reported cases in India

We can see that the **age group < 40 is the most affected,** which is against the trend that says elderly people are more at risk of being affected. Only 17% of people >60 are affected.

Gender-wise Disease Spread Analysis

```
labels = ['Male', 'Female']
sizes = []
sizes.append(list(individual_details['gender'].value_counts())[0])
sizes.append(list(individual_details['gender'].value_counts())[1])

explode = (0.1, 0)
colors = ['#66b3ff','#ff9999']

plt.figure(figsize= (12,7))
plt.pie(sizes, explode=explode, labels=labels, colors=colors, autopct='%1.1f%%',
        shadow=True, startangle=90)

plt.title('Percentage of Gender (Ignoring the Missing Values)',fontsize = 20)
plt.axis('equal')
plt.tight_layout()
```

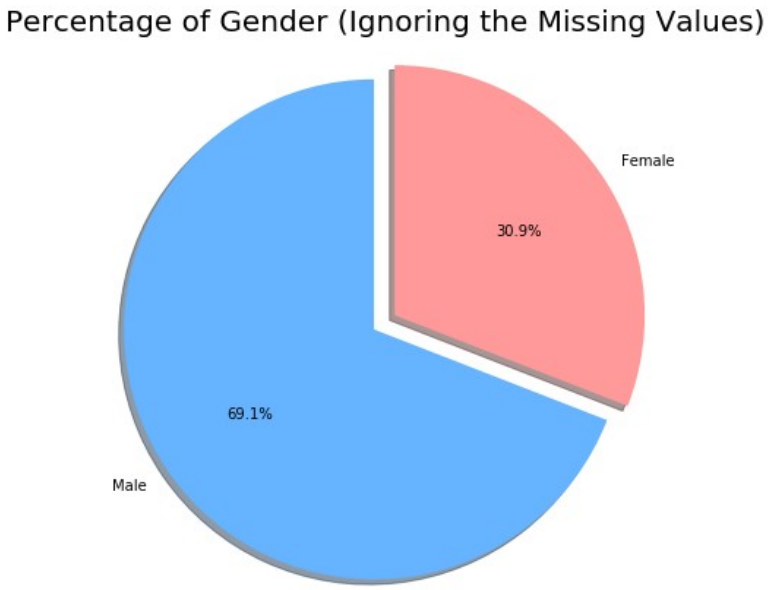

Fig. 8. Gender-wise distribution of cases in India

According to above analysis, Men are the most affected accounting to 70%. But, remember we have ~20% data missing.

The Spike in India

Here, we try to understand trend in active, recovered and expired cases in India. We group the data by Country in the international dataset and plot the Indian data points on the graph as shown below.

```
df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
df2 = deaths_df.groupby('Country/Region').sum().reset_index()
df3 = recovered_df.groupby('Country/Region').sum().reset_index()

k = df1[df1['Country/Region']=='India'].loc[:,'1/30/20':]
india_confirmed = k.values.tolist()[0]

k = df2[df2['Country/Region']=='India'].loc[:,'1/30/20':]
india_deaths = k.values.tolist()[0]

k = df3[df3['Country/Region']=='India'].loc[:,'1/30/20':]
india_recovered = k.values.tolist()[0]

plt.figure(figsize= (12,6))
plt.xticks(rotation = 90 ,fontsize = 11)
plt.yticks(fontsize = 10)
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Total Confirmed, Active, Death in India" , fontsize = 20)

ax1 = plt.plot_date(y= india_confirmed,x= dates_india,label = 'Confirmed',linestyle ='-',color = 'b')
ax2 = plt.plot_date(y= india_recovered,x= dates_india,label = 'Recovered',linestyle ='-',color = 'g')
ax3 = plt.plot_date(y= india_deaths,x= dates_india,label = 'Death',linestyle ='-',color = 'r')
plt.legend();
```

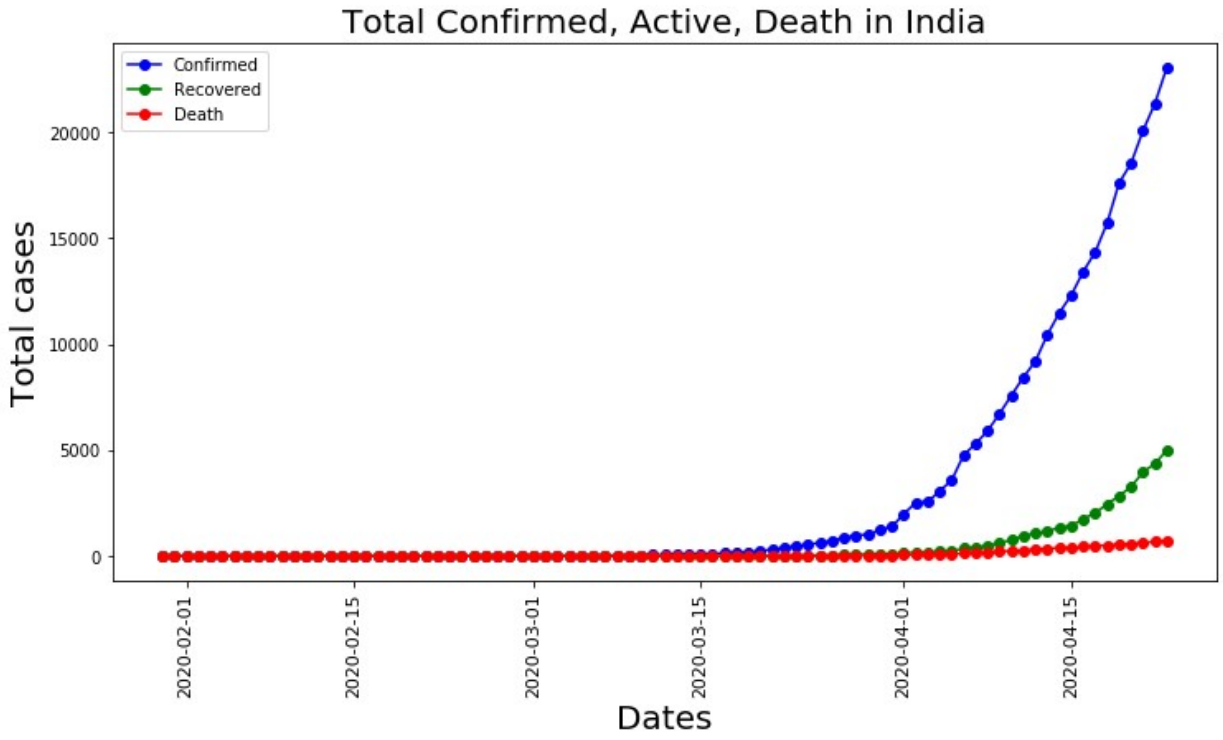

Fig. 9. In the figure, according to our data analysis the blue line represents the confirmed cases of COVID-19 in India, the green line represents the recovered cases, and the red line represents the number of deaths due to the pandemic.

Visualizing on Indian map using geopandas:

First, we need to store the Indian map in a geopandas file. We also need to rename the columns of the stored file to the actual column name of states according to our dataset. This is important because geopandas will not recognize the data to plot if the names mismatch.

```
map_data = gpd.read_file('../MajorProject/DataIndia/IndianMap/Indian_States.shp')
map_data.rename(columns = {'st_nm':'Name of State / UT'}, inplace = True)
map_data.tail()
```

	Name of State / UT	geometry
31	Uttar Pradesh	MULTIPOLYGON (((80.44802 24.99631, 80.44080 24...
32	Uttarakhand	POLYGON ((79.21047 31.34846, 79.21386 31.34680...
33	West Bengal	MULTIPOLYGON (((88.01861 21.57278, 88.01889 21...
34	Odisha	MULTIPOLYGON (((86.38937 19.96351, 86.38840 19...
35	Andhra Pradesh	MULTIPOLYGON (((81.10380 17.82269, 81.10610 17...

We also need to change the names of states in geopandas file to match the names of states and UT in the dataset. After this, we will merge the geopandas file to the grouped_by_state dataset we created before.

```
map_data['Name of State / UT'] = map_data['Name of State / UT'].str.replace("&","and")
map_data['Name of State / UT'].replace('Arunanchal Pradesh',
                                'Arunachal Pradesh', inplace = True)
map_data['Name of State / UT'].replace('Telangana',
                                'Telengana', inplace = True)
map_data['Name of State / UT'].replace('NCT of Delhi',
                                'Delhi', inplace = True)
map_data['Name of State / UT'].replace('Andaman and Nicobar Island',
                                'Andaman and Nicobar Islands',inplace = True)
```

```
merged_data = pd.merge(map_data, grouped_by_state,
                       how = 'left', on = 'Name of State / UT')
merged_data.fillna(0, inplace = True)
merged_data.head()
```

Below is the final dataframe that we will use to visualize the total number of confirmed cases in India.

	Name of State / UT	geometry	Date	Cured/Discharged/Migrated	Death	Total Confirmed cases
0	Andaman and Nicobar Islands	MULTIPOLYGON (((93.71976 7.20707, 93.71909 7.2...	2020-04-30 00:00:00	15.0	0.0	33.0
1	Arunachal Pradesh	POLYGON ((96.16261 29.38078, 96.16860 29.37432...	2020-04-30 00:00:00	1.0	0.0	1.0
2	Assam	MULTIPOLYGON (((89.74323 26.30362, 89.74290 26...	2020-04-30 00:00:00	29.0	1.0	38.0
3	Bihar	MULTIPOLYGON (((84.50720 24.26323, 84.50355 24...	2020-04-30 00:00:00	65.0	2.0	392.0
4	Chandigarh	POLYGON ((76.84147 30.75996, 76.83599 30.73623...	2020-04-30 00:00:00	17.0	0.0	56.0

Now we plot the data on the map to get a view of the spread of the virus in India.

```
fig, ax = plt.subplots(1, figsize=(20, 12))
ax.axis('off')
ax.set_title('Covid-19 Statewise Data – Confirmed Cases',
             fontdict =  {'fontsize': '25', 'fontweight' : '3'})
merged_data.plot(column = 'Total Confirmed cases', cmap = 'Reds',
                 linewidth=0.8, ax=ax, edgecolor='0.8', legend = True)
plt.show()
```

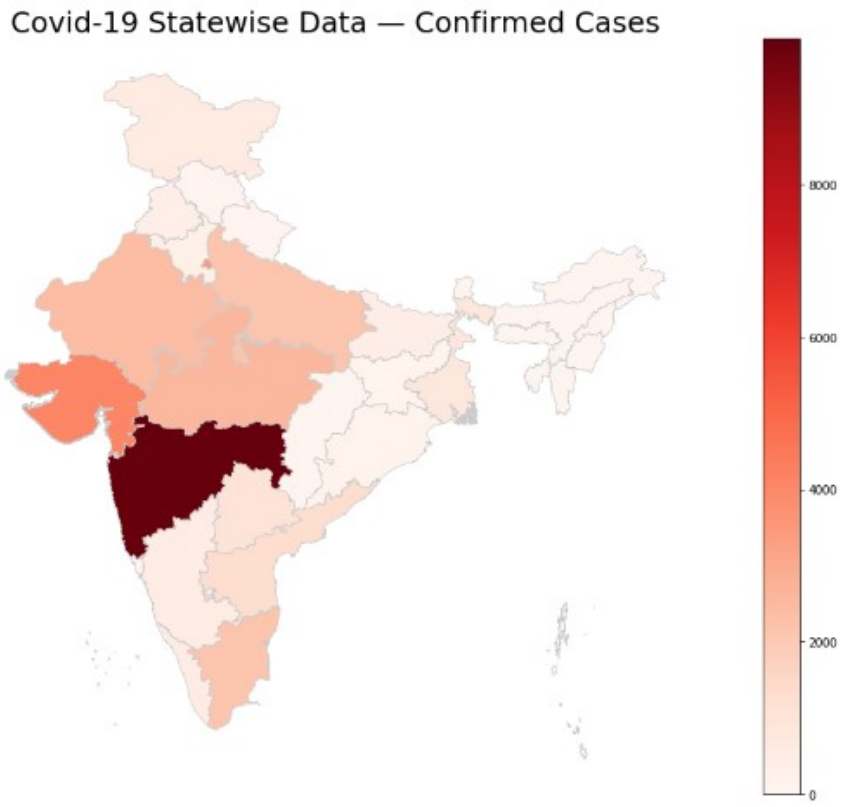

Fig. 10. Heatmap of the number of reported cases by states.

ICMR Tests Analysis

```
ICMR_details['Percent_positive'] =
round((ICMR_details['TotalPositiveCases']/ICMR_details['TotalSamplesTested'])*100,1)

fig, ax1 = plt.subplots(figsize= (15,5))
ax1.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
ax1.set_ylabel('Positive Cases (% of Total Samples Tested)')
ax1.bar(ICMR_details['DateTime'] , ICMR_details['Percent_positive'],
        color="red",label = 'Percentage of Positive Cases')
ax1.text(ICMR_details['DateTime'][0],4,
         'Total Samples Tested as of Apr 19th = 401586', style='italic',fontsize= 10,
        bbox={'facecolor': 'white' ,'alpha': 0.5, 'pad': 5})

ax2 = ax1.twinx()
ax2.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
ax2.set_ylabel('Num Samples Tested')
ax2.fill_between(ICMR_details['DateTime'],ICMR_details['TotalSamplesTested'],
                 color = 'black',alpha = 0.5,
                 label = 'Samples Tested');

plt.legend(loc="upper left")
plt.title('Total Samples Tested')
plt.show()
```

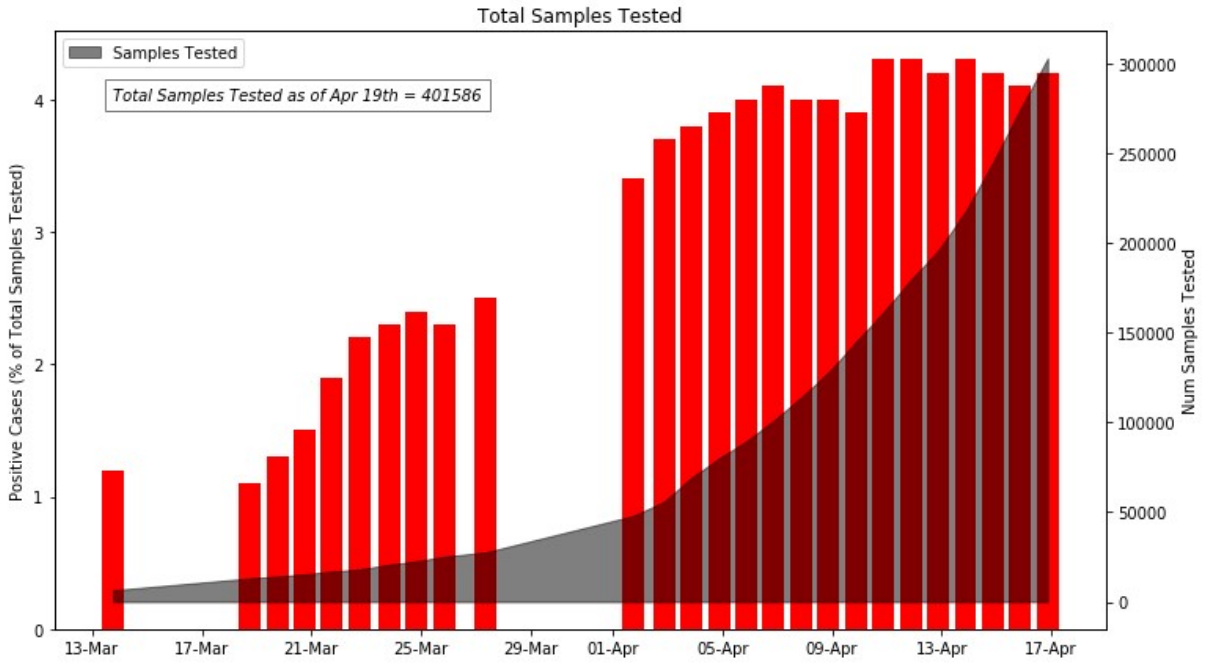

Fig. 11. In the above plot, the red bars show the number of samples tested daily, the shaded dark region show the number of positive cases.

Statewise Insights

Now, we analyze the data state-wise to see if we can extract some useful information out of it.

```
state_cases = india_covid_19.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.sort_values('Confirmed', ascending= False).fillna(0).style.background_gradient
(cmap='Blues',subset=["Confirmed"])\
                        .background_gradient(cmap='Blues',subset=["Deaths"])\
                        .background_gradient(cmap='Blues',subset=["Cured"])\
                        .background_gradient(cmap='Blues',subset=["Active"])\
                        .background_gradient(cmap='Blues',subset=["Death Rate (per 100)"])\
                        .background_gradient(cmap='Blues',subset=["Cure Rate (per 100)"])
```

We clean, process and organize the data to plot the state-wise curve. We remove all the columns with unassigned values. We also remove the Nagaland row from the dataset

because there is no reported case from there. The zero value in the cell can cause problems so it is better to remove it.

```
all_state = list(india_covid_19['State/UnionTerritory'].unique())
all_state.remove('Unassigned')
#all_state.remove('Nagaland#')
#all_state.remove('Nagaland')
latest = india_covid_19[india_covid_19['Date'] > '24-03-20']
state_cases = latest.groupby('State/UnionTerritory')['Confirmed','Deaths','Cured'].max().reset_index()
latest['Active'] = latest['Confirmed'] - (latest['Deaths']- latest['Cured'])
state_cases = state_cases.sort_values('Confirmed', ascending= False).fillna(0)
states =list(state_cases['State/UnionTerritory'][0:15])
states_confirmed = {}
states_deaths = {}
states_recovered = {}
states_active = {}
states_dates = {}

for state in states:
    df = latest[latest['State/UnionTerritory'] == state].reset_index()
    k = []
    l = []
    m = []
    n = []
    for i in range(1,len(df)):
        k.append(df['Confirmed'][i]-df['Confirmed'][i-1])
        l.append(df['Deaths'][i]-df['Deaths'][i-1])
        m.append(df['Cured'][i]-df['Cured'][i-1])
        n.append(df['Active'][i]-df['Active'][i-1])
    states_confirmed[state] = k
    states_deaths[state] = l
    states_recovered[state] = m
```

Now, we plot the data using matplotlib.

```
def calc_movingaverage(values ,N):
    cumsum, moving_aves = [0], [0,0]
    for i, x in enumerate(values, 1):
        cumsum.append(cumsum[i-1] + x)
        if i>=N:
            moving_ave = (cumsum[i] - cumsum[i-N])/N
            moving_aves.append(moving_ave)
    return moving_aves

fig = plt.figure(figsize= (25,17))
plt.suptitle('5-Day Moving Average of Confirmed Cases in Top 15 States',fontsize = 20,y=1.0)
k=0
for i in range(1,15):
    ax = fig.add_subplot(5,3,i)
    ax.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
    ax.bar(states_dates[states[k]],states_confirmed[states[k]],label = 'Day wise Confirmed Cases ')
    moving_aves = calc_movingaverage(states_confirmed[states[k]],5)
    ax.plot(states_dates[states[k]][:-2],moving_aves,color='red',label = 'Moving Average',linewidth =3)
    plt.title(states[k],fontsize = 20)
    handles, labels = ax.get_legend_handles_labels()
    fig.legend(handles, labels, loc='upper left')
    k=k+1
plt.tight_layout(pad=3.0)
```

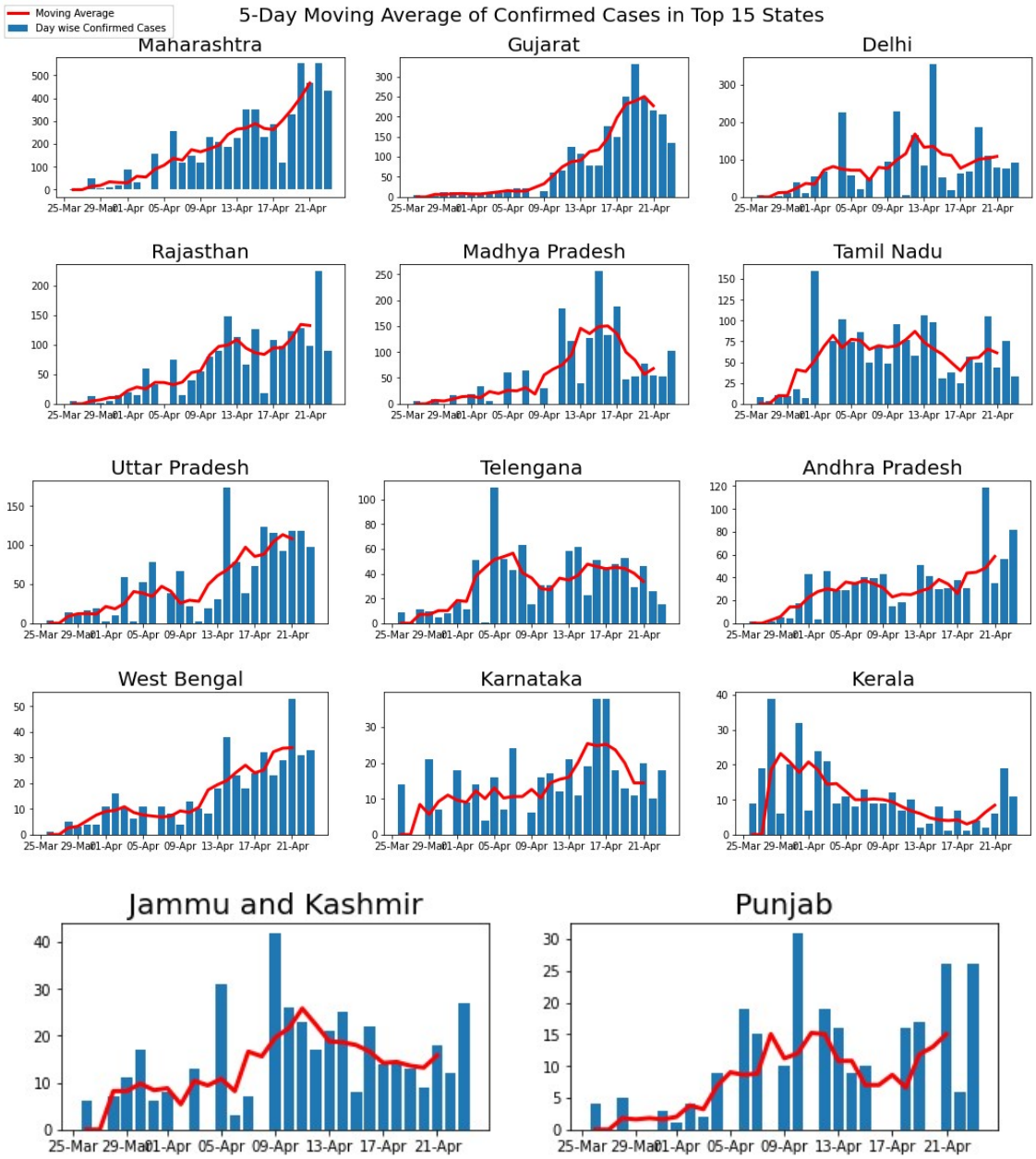

Fig. 12. The above figure shows the number of cases daily in various Indian states. The red line plot shows the moving average of the cases in respective states.

Worrying: Maharashtra, Gujarat, UP, Rajasthan, WB

Decent Recovery: Kerala, Haryana, Karnataka

Others: Uncertain

Healthcare Insights

Now, we analyze the state-wise data for the number of health care facilities at state and district level.

```
top_10_primary = hospital_beds.nlargest(10,'NumPrimaryHealthCenters_HMIS')
top_10_community = hospital_beds.nlargest(10,'NumCommunityHealthCenters_HMIS')
top_10_district_hospitals = hospital_beds.nlargest(10,'NumDistrictHospitals_HMIS')
top_10_public_facility = hospital_beds.nlargest(10,'TotalPublicHealthFacilities_HMIS')
top_10_public_beds = hospital_beds.nlargest(10,'NumPublicBeds_HMIS')

plt.figure(figsize=(12,7))
plt.suptitle('Top 10 States in each Health Facility',fontsize=20)
plt.subplot(221)
plt.title('Primary Health Centers')
plt.barh(top_10_primary['State/UT'],top_10_primary['NumPrimaryHealthCenters_HMIS'],color ='#87479d');

plt.subplot(222)
plt.title('Community Health Centers')
plt.barh(top_10_community['State/UT'],top_10_community['NumCommunityHealthCenters_HMIS'],color = '#9370db');

plt.subplot(224)
plt.title('Total Public Health Facilities')
plt.barh(top_10_community['State/UT'],top_10_public_facility['TotalPublicHealthFacilities_HMIS'],color='#9370db');

plt.subplot(223)
plt.title('District Hospitals')
plt.barh(top_10_community['State/UT'],top_10_district_hospitals['NumDistrictHospitals_HMIS'],color = '#87479d');
```

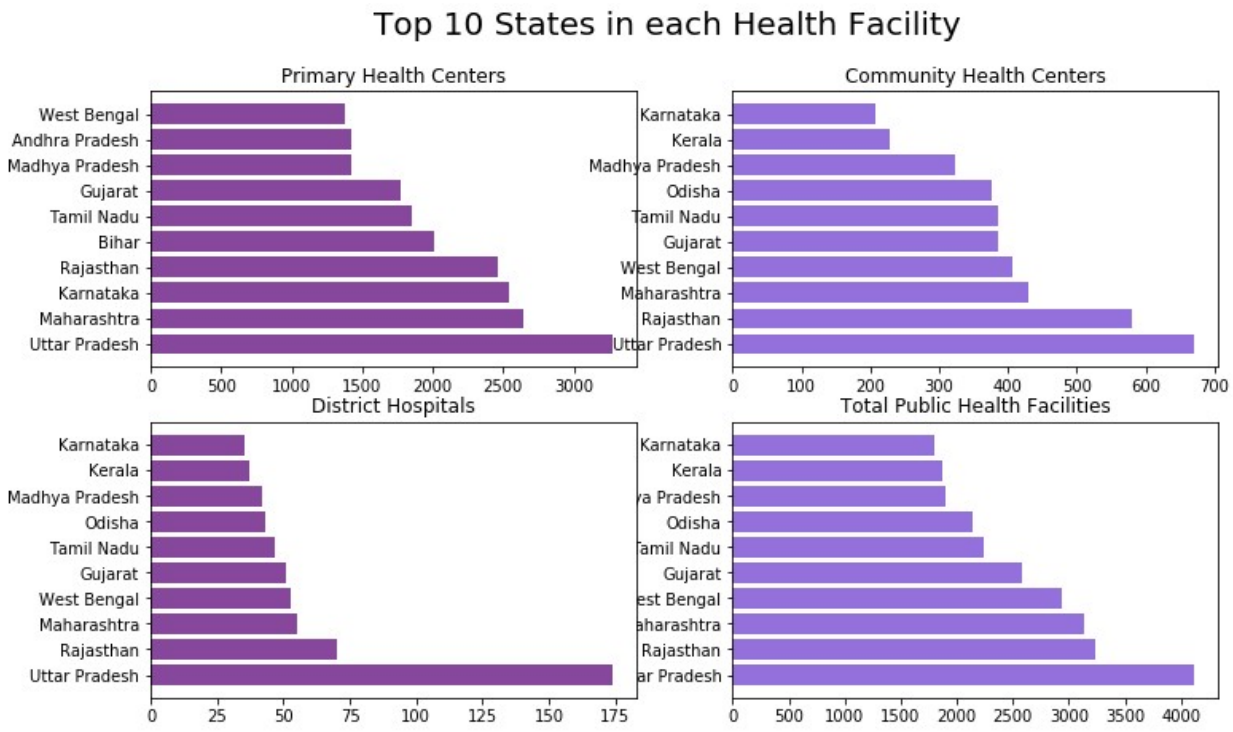

Fig. 13. Above figure shows the number of healthcare related facilities available in each state of in India.

Here is the analysis of data in rural and urban areas on the basis of no. of available beds for patients.

```
top_rural_hos = hospital_beds.nlargest(10,'NumRuralHospitals_NHP18')
top_rural_beds = hospital_beds.nlargest(10,'NumRuralBeds_NHP18')
top_urban_hos = hospital_beds.nlargest(10,'NumUrbanHospitals_NHP18')
top_urban_beds = hospital_beds.nlargest(10,'NumUrbanBeds_NHP18')

plt.figure(figsize=(12,7))
plt.suptitle('Urban and Rural Health Facility',fontsize=20)
plt.subplot(221)
plt.title('Rural Hospitals')
plt.barh(top_rural_hos['State/UT'],top_rural_hos['NumRuralHospitals_NHP18'],color = '#87479d');

plt.subplot(222)
plt.title('Urban Hospitals')
plt.barh(top_urban_hos['State/UT'],top_urban_hos['NumUrbanHospitals_NHP18'],color = '#9370db');

plt.subplot(223)
plt.title('Rural Beds')
plt.barh(top_rural_beds['State/UT'],top_rural_beds['NumRuralBeds_NHP18'],color = '#87479d');

plt.subplot(224)
plt.title('Urban Beds')
plt.barh(top_urban_beds['State/UT'],top_urban_beds['NumUrbanBeds_NHP18'],color = '#9370db');
```

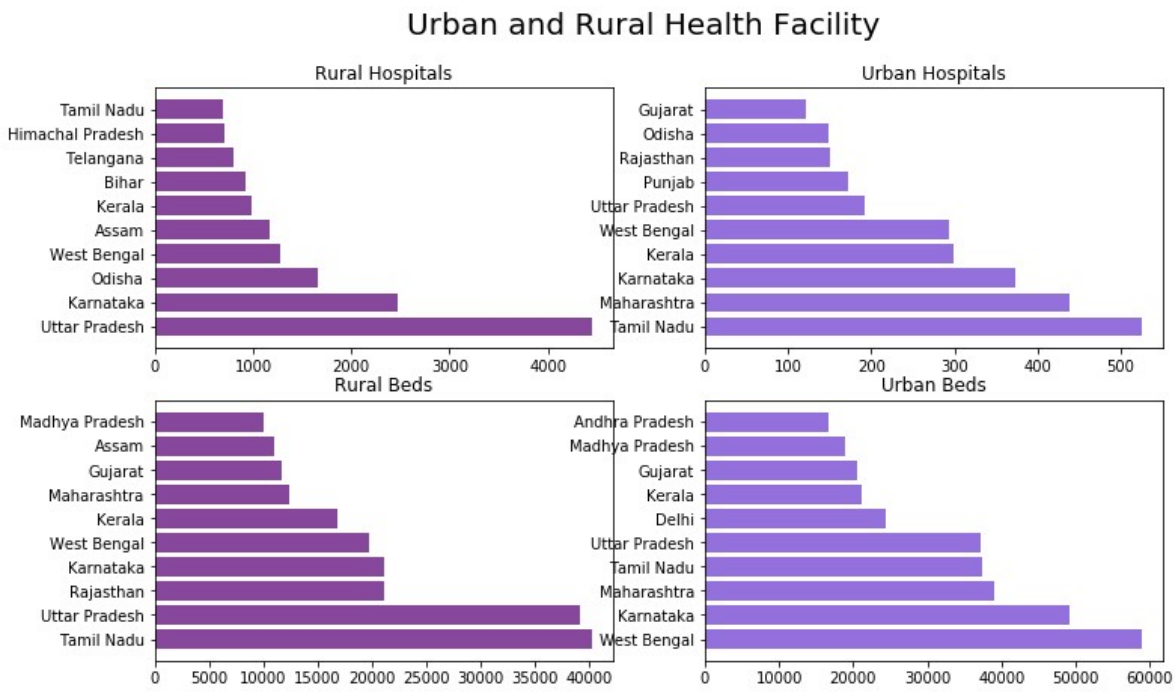

Fig. 14. In the above figure, the difference in the number of available healthcare facilities in rural and urban areas is depicted.

Statewise Testing

ICMR Testing Labs:

In this section, we understand the testing labs distribution for Covid-19 all across India.

```
values = list(ICMR_labs['state'].value_counts())
names = list(ICMR_labs['state'].value_counts().index)

plt.figure(figsize=(15,10))
sns.set_color_codes("pastel")
plt.title('ICMR Testing Centers in each State', fontsize = 20)
sns.barplot(x= values, y= names,color = '#9370db');
```

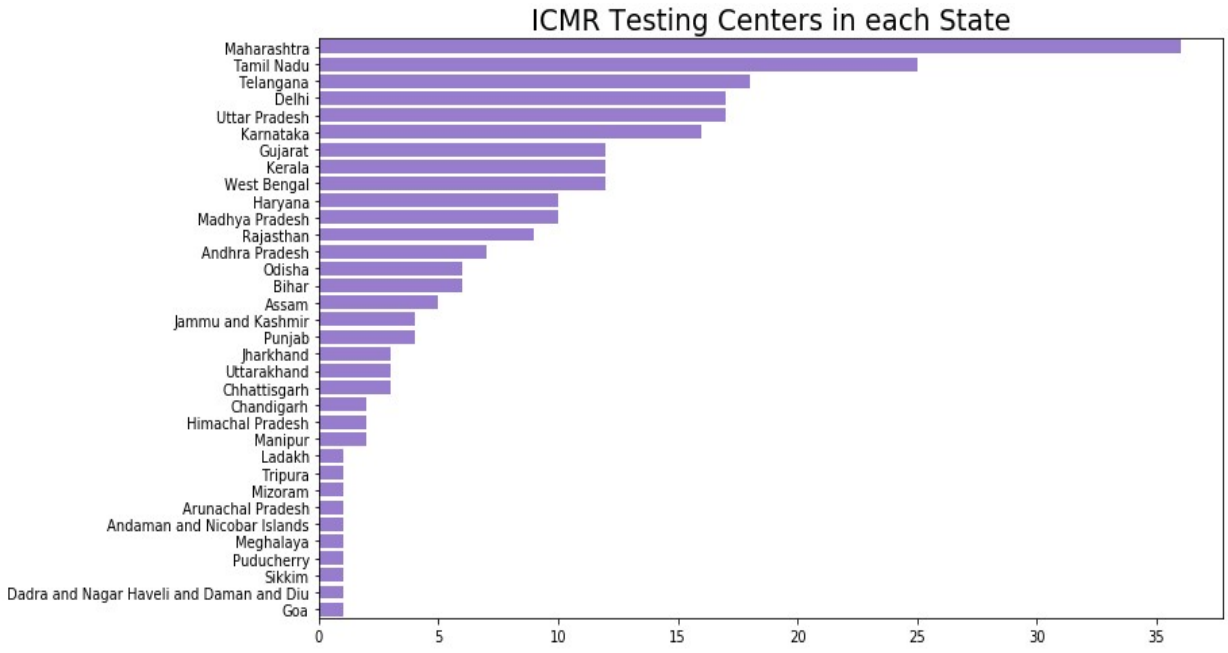

Fig. 15. ICMR testing centers in each state.

ICMR Tests Results:

We analyze the ICMR testing data state-wise and collect information about positive and negative cases in each state.

```
state_test = pd.pivot_table(state_testing, values=['TotalSamples','Negative','Positive'], index='State', aggfunc='max')
state_names = list(state_test.index)
state_test['State'] = state_names

plt.figure(figsize=(15,10))
sns.set_color_codes("pastel")
sns.barplot(x="TotalSamples", y= state_names, data=state_test,label="Total Samples", color = '#9370db')
sns.barplot(x='Negative', y=state_names, data=state_test,label='Negative', color= '#ff9999')
sns.barplot(x='Positive', y=state_names, data=state_test,label='Positive', color='#87479d')
plt.title('Testing statewise insight',fontsize = 20)
plt.legend(ncol=2, loc="lower right", frameon=True);
```

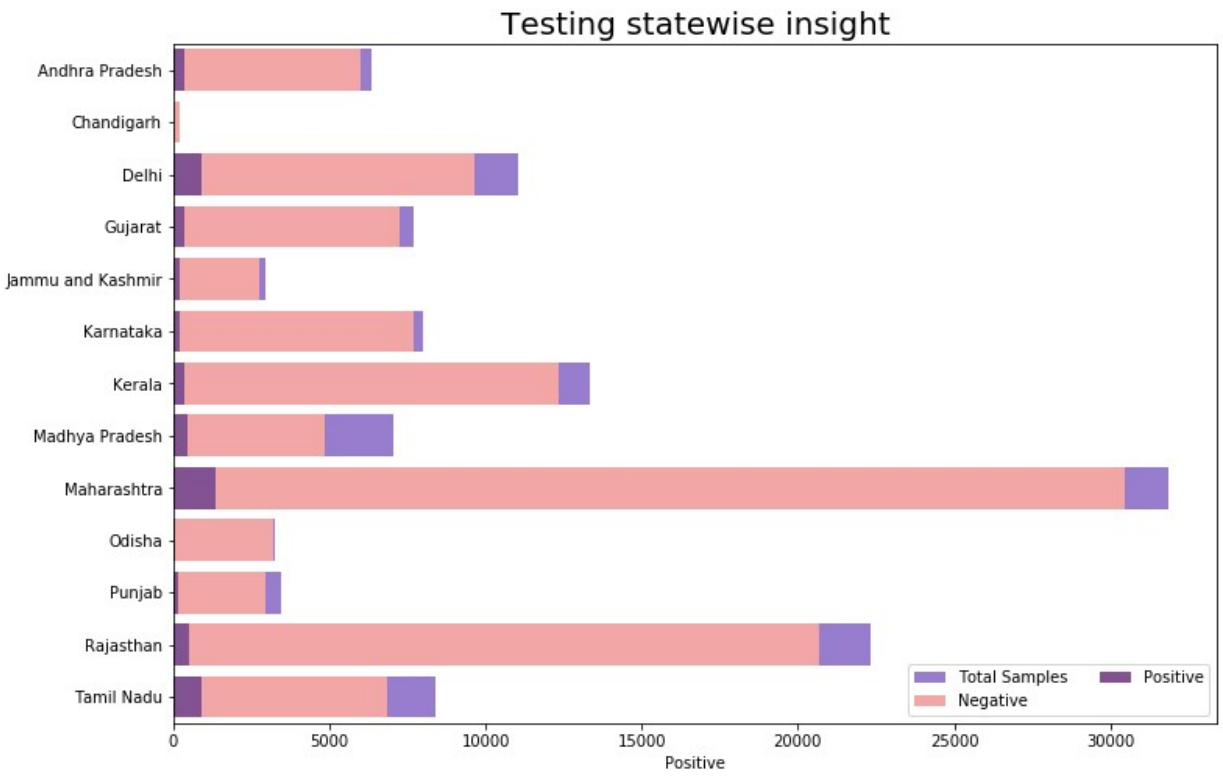

Fig. 16. Statewise insights in the number of testing samples and the percentage of positive cases in various states.

- **Understanding the Covid-19 impact on world**

Analyzing data for Covid-19 cases all over the world.

```
world_confirmed = confirmed_df[confirmed_df.columns[-1:]].sum()
world_recovered = recovered_df[recovered_df.columns[-1:]].sum()
world_deaths = deaths_df[deaths_df.columns[-1:]].sum()
world_active = world_confirmed - (world_recovered - world_deaths)

labels = ['Active','Recovered','Deceased']
sizes = [world_active,world_recovered,world_deaths]
color= ['#66b3ff','green','red']
explode = []

for i in labels:
    explode.append(0.05)

plt.figure(figsize= (12,7))
plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=9, explode =explode,colors = color)
centre_circle = plt.Circle((0,0),0.70,fc='white')

fig = plt.gcf()
fig.gca().add_artist(centre_circle)
plt.title('World COVID-19 Cases',fontsize = 20)
plt.axis('equal')
plt.tight_layout()
```

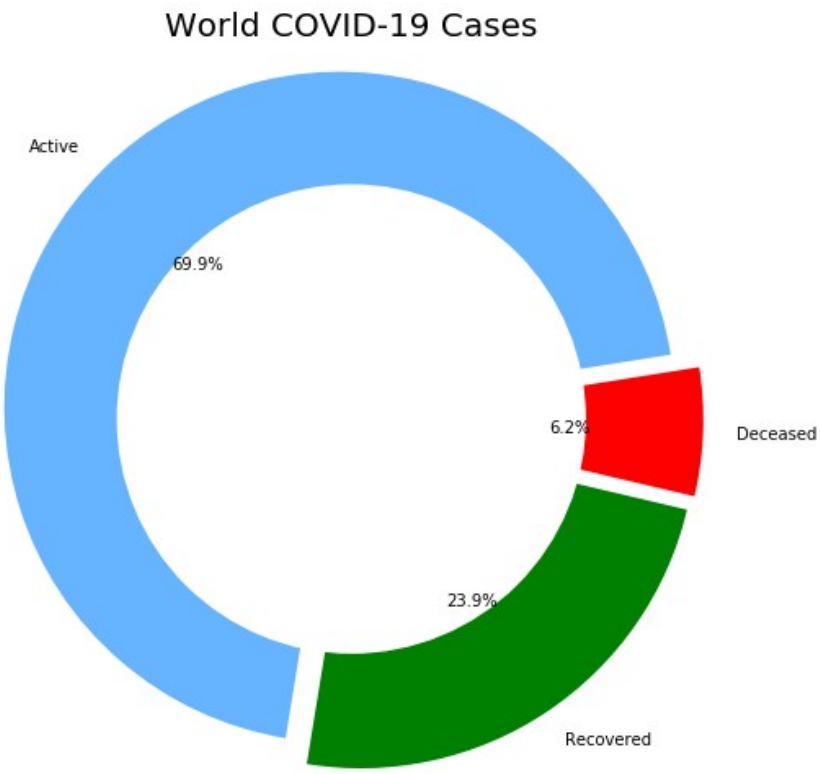

Fig. 17. Distribution of the number of total cases of COVID-19 across the world.

Analyzing data for countries that are classified as hotspots.

```
hotspots = ['China','Germany','Iran','Italy','Spain','US','Korea, South',
            'France','Turkey','United Kingdom','India']
dates = list(confirmed_df.columns[4:])
dates = list(pd.to_datetime(dates))
dates_india = dates[8:]

df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
df2 = deaths_df.groupby('Country/Region').sum().reset_index()
df3 = recovered_df.groupby('Country/Region').sum().reset_index()

global_confirmed = {}
global_deaths = {}
global_recovered = {}
global_active= {}

for country in hotspots:
    k =df1[df1['Country/Region'] == country].loc[:,'1/30/20':]
    global_confirmed[country] = k.values.tolist()[0]

    k =df2[df2['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths[country] = k.values.tolist()[0]

    k =df3[df3['Country/Region'] == country].loc[:,'1/30/20':]
    global_recovered[country] = k.values.tolist()[0]
```

```
for country in hotspots:
    k = list(map(int.__sub__, global_confirmed[country], global_deaths[country]))
    global_active[country] = list(map(int.__sub__, k, global_recovered[country]))

fig = plt.figure(figsize= (15,15))
plt.suptitle('Active, Recovered, Deaths in Hotspot Countries and India as of April 20',
             fontsize = 20,y=1.0)
#plt.legend()
k=0
for i in range(1,12):
    ax1 = fig.add_subplot(6,2,i)
    ax1.xaxis.set_major_formatter(mdates.DateFormatter('%d-%b'))
    ax2.bar(dates_india,global_active[hotspots[k]],color = 'green',alpha = 0.6,label = 'Active');
    ax2.bar(dates_india,global_recovered[hotspots[k]],color='grey',label = 'Recovered');
    ax2.bar(dates_india,global_deaths[hotspots[k]],color='red',label = 'Death');
    plt.title(hotspots[k])
    handles, labels = ax2.get_legend_handles_labels()
    fig.legend(handles, labels, loc='upper left')
    k=k+1

plt.tight_layout(pad=3.0)
```

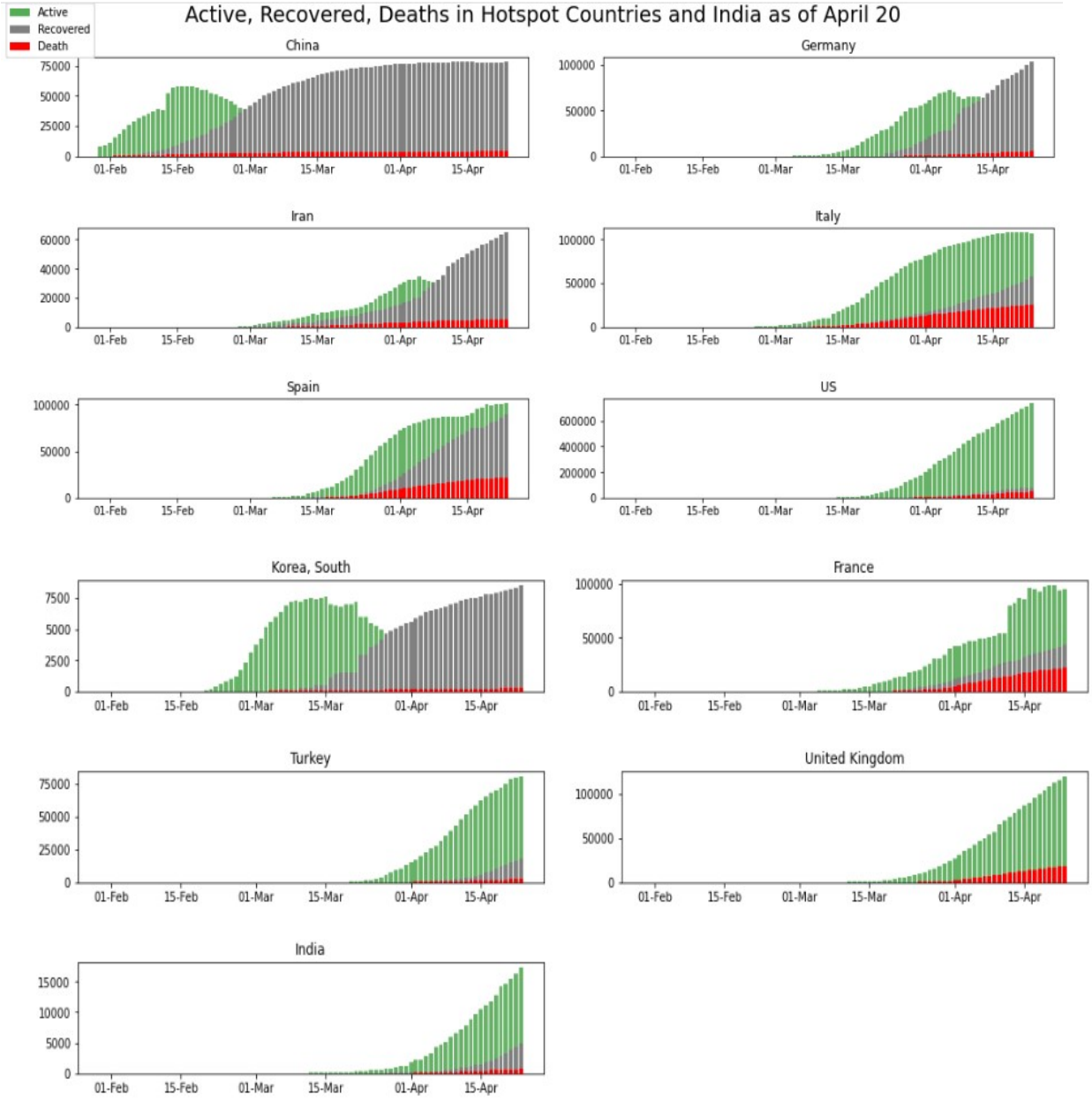

Fig. 18. Country-wise distribution of the cases in various countries.

Certain Recovery : South Korea, Germany, Iran (Flattened the Curve!)

Uncertain Recovery : Italy, France, Spain (Flattening!)

United Kingdom is showing very few recoveries.

For Hotspot Countries, deaths per million.

```
hotspots = ['China','Germany','Iran','Italy','Spain','United States','South Korea',
            'France','Turkey','United Kingdom','India']
country_death_rate = pd.DataFrame(columns = ['country','day1','day2','day3'])
world_population['Population (2020)'] = world_population['Population (2020)']/1000000

d1=[]
d2 =[]
d3 = []
for country in hotspots:
    p = float(world_population[world_population['Country (or dependency)'] == country ]
              ['Population (2020)'])
    if country == 'United States':
        k = global_deaths['US'][-3:]
    elif country == 'South Korea':
        k = global_deaths['Korea, South'][-3:]
    else:
        k = global_deaths[country][-3:]
    d1.append(round(k[0]/p,2))
    d2.append(round(k[1]/p,2))
    d3.append(round(k[2]/p,2))

country_death_rate['country'] = hotspots
country_death_rate['day1'] = d1
country_death_rate['day2'] = d2
country_death_rate['day3'] = d3
```

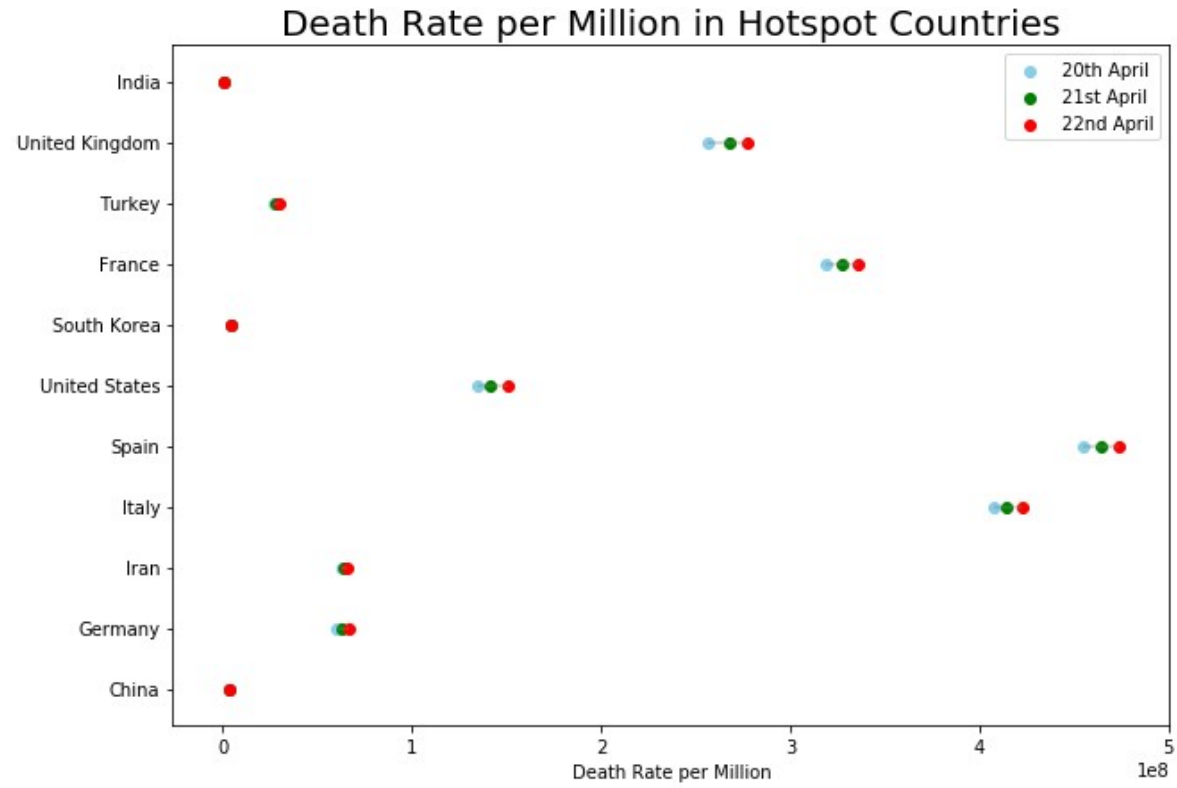

Fig. 19. Number of deaths per million in the worst affected countries.

- **Comparison of Covid-19 cases in India with other countries**

```
countries = ['China','US', 'Italy', 'Spain', 'France','India']

global_confirmed = []
global_recovered = []
global_deaths = []
global_active = []

for country in countries:
    k =df1[df1['Country/Region'] == country].loc[:,'1/30/20':]
    global_confirmed.append(k.values.tolist()[0])

    k =df2[df2['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths.append(k.values.tolist()[0])

    k =df3[df3['Country/Region'] == country].loc[:,'1/30/20':]
    global_deaths.append(k.values.tolist()[0])

plt.figure(figsize= (12,7))
plt.xticks(rotation = 90 ,fontsize = 11)
plt.yticks(fontsize = 10)
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Comparison with other Countries" , fontsize = 20)

for i in range(len(countries)):
    plt.plot_date(y= global_confirmed[i],x= dates_india,label = countries[i],linestyle ='-')
plt.legend();
```

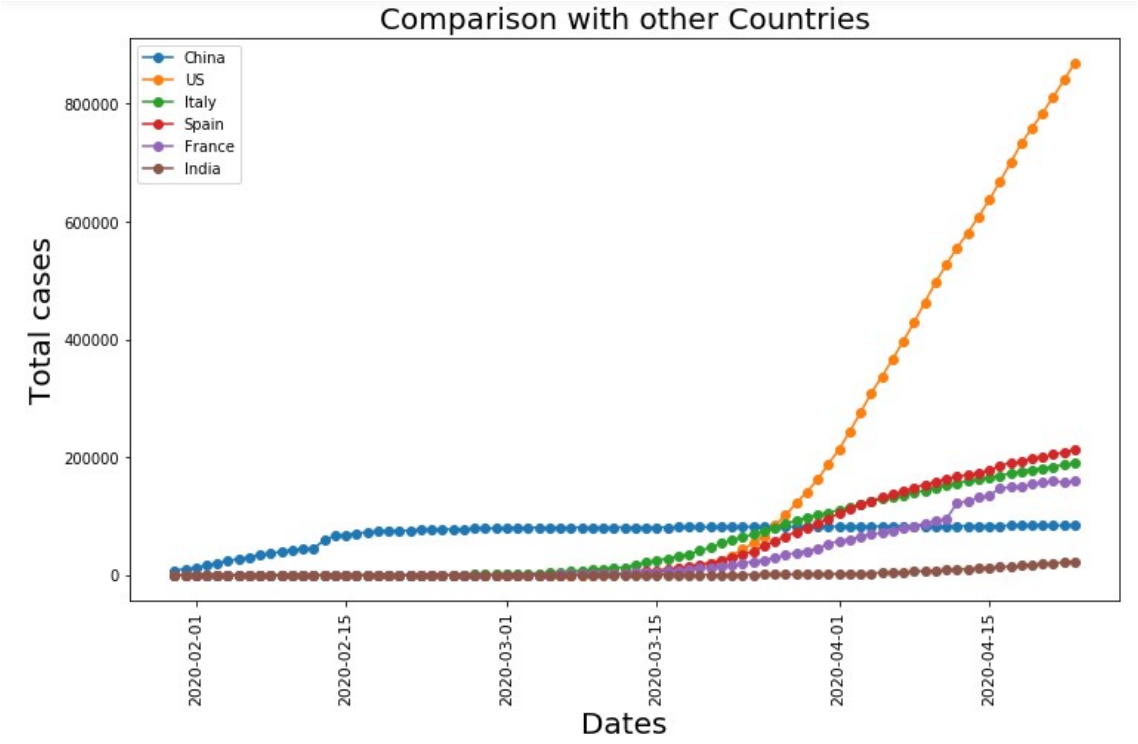

Fig. 20. Above figure shows the line plots of various countries

Though being highly populated the relative number of confirmed cases of India is low compared to other countries.

This could be because of two reasons :

- 21 day lockdown and another 19 day lockdowns imposed by prime minister Narendra Modi all across India.

  (Source : Health Ministry of India).

- Low testing rate – Some experts are attributing the low number of cases to the low testing rate of the virus in India.

  (Source: news18)

Results of Lockdown in India :

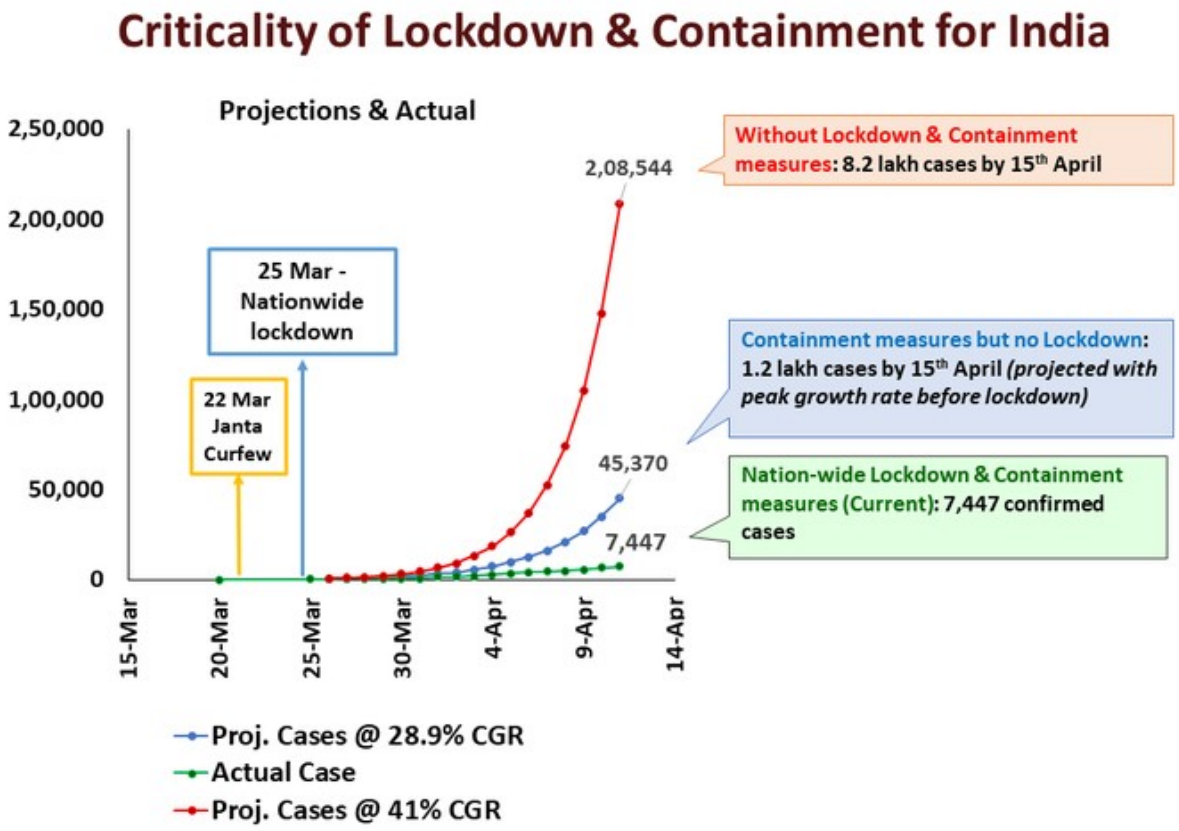

Fig. 21. Above figure shows the predicted cases in different scenarios by various news websites (source: BBCIndia).

4.3 Data Preprocessing

In the domain of machine learning, one of the most important steps is considered to be data preprocessing because your model and ultimately predictions from the model is going to be as good as the quality of the data you provide. The techniques of data collecting are inherently unreliable because they invariably contain categorical features which proves to be very hard for the machine learning models to process. They might contain missing values because of a various number of reasons. Therefore, it is very important to analyze and understand the data as much as we can. Data preprocessing should be give the most amount of time in a project. This is considered to be one of the best practice in the machine learning community. The presence of unreliable, redundant and noisy data results in inaccurate outputs and can consume long periods of time in knowledge discovery.

4.3.1 Data Transformations

First, we need to change the Date column in our datasets to pandas datetime format. This will make it easy for us to plot the graphs using the data and easy for the models to analyze the data.

```
india_covid_19['Date'] = pd.to_datetime(india_covid_19['Date'],dayfirst = True)
state_testing['Date'] = pd.to_datetime(state_testing['Date'])
ICMR_details['DateTime'] = pd.to_datetime(ICMR_details['DateTime'],dayfirst = True)
ICMR_details = ICMR_details.dropna(subset=['TotalSamplesTested', 'TotalPositiveCases'])
```

Now, we create individual numpy arrays for each component of the datetime field in our test and train data. This trick will help us to organize the datasets in a way that will help us to create suitable columns which will facilitate the models which we use them on.

```
train['day'] = train['Date'].dt.day
train['month'] = train['Date'].dt.month
train['dayofweek'] = train['Date'].dt.dayofweek
train['dayofyear'] = train['Date'].dt.dayofyear
train['quarter'] = train['Date'].dt.quarter
train['weekofyear'] = train['Date'].dt.weekofyear
test['day'] = test['Date'].dt.day
test['month'] = test['Date'].dt.month
test['dayofweek'] = test['Date'].dt.dayofweek
test['dayofyear'] = test['Date'].dt.dayofyear
test['quarter'] = test['Date'].dt.quarter
test['weekofyear'] = test['Date'].dt.weekofyear
```

It is very important to remove redundant information from the datasets. Hence, cleaning the datasets to reduce computational time and sometimes inaccurate results is very important.

```
countries = list(train['Country_Region'].unique())
india_code = countries.index('India')
train = train.drop(['Date','Id'],1)
test =  test.drop(['Date'],1)
```

We noticed earlier that the province_state attribute has some missing values. Most machine learning algorithms can't work with datasets that have missing values.

There are 3 ways to solve this problem:

1. We could remove the whole attribute.

2. We could get rid of the districts that contain missing values.

3. We could replace them with zeroes, the median or the mean.

We choose option three. Sklearn provides you with "Encoders" to do this. We first need to specify an encoder instance, that specifies that we want to replace each attribute's non-compatible values with another value of our choosing.

Here, we will use OrdinalEncoder() from the sklearn module. Since our model will not be able to handle categorical features, we need to transform the categorical column with some integer value. Therefore, the ordinal encoder takes the categorical ffr=eatures column as a numpy array and fills the ordinal values in place of them. Now, our model will be comfortable in dealing with this data.

```
train.Province_State.fillna('NaN', inplace=True)
oe = OrdinalEncoder()
train[['Province_State','Country_Region']] =
oe.fit_transform(train.loc[:,['Province_State','Country_Region']])

test.Province_State.fillna('NaN', inplace=True)
oe = OrdinalEncoder()
test[['Province_State','Country_Region']] =
oe.fit_transform(test.loc[:,['Province_State','Country_Region']])
```

4.3.2 Train/Test Split

It is important that we now set a part of the data aside. Your brain is an amazing pattern detection system and therefore extremely prune to over-fitting. If you would also look at the test set during the visual explanation, which we will do now, you may find some patterns unconsciously within the data, which let you select a particular algorithm that leads to an over-fitted model. Because of that you could launch a system that performs not as well on new data than expected. We use the straight forward "train_test_split()" method from sklearn to split our data into train and test subsets.
In our case we split the data into 80% training data and the rest into validation data.

Now, we also need to randomize our data to avoid over-fitting the data.

```
columns = ['day','month','dayofweek','dayofyear','quarter','weekofyear','Province_State',
           'Country_Region','ConfirmedCases','Fatalities']
test_columns = ['day','month','dayofweek','dayofyear','quarter','weekofyear',
                'Province_State','Country_Region']
train = train[columns]
x = train.drop(['Fatalities','ConfirmedCases'], 1)
y = train['ConfirmedCases']
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
test = test[test_columns]
test_india = test[test['Country_Region'] == india_code]
```

Finally, after all the preprocessing and transformation, our data looks like following :-

- **Input Data**

```
x_train.head()
```

	day	month	dayofweek	dayofyear	quarter	weekofyear	Province_State	Country_Region
8157	3	4	4	94	2	14	NaN	El Salvador
8581	25	2	1	56	1	9	NaN	Fiji
22391	23	3	0	83	1	13	Bermuda	United Kingdom
8783	27	1	0	27	1	5	French Polynesia	France
6498	21	2	4	52	1	8	Xinjiang	China

- **Output Data**

```
y_train.head()
```

```
8157     46.0
8581      0.0
22391     6.0
8783      0.0
6498     76.0
Name: ConfirmedCases, dtype: float64
```

4.4 Forecasting and Prediction

4.4.1 Prediction using growth factor (basic mathematical model)

Calculation of the growth factor: Growth factor represents the increase in the number of cases daily. It is calculated by averaging the differences of the number of cases on a particular day with the previous day.

This calculation is as follows:-

```
df1 = confirmed_df.groupby('Country/Region').sum().reset_index()
df2 = deaths_df.groupby('Country/Region').sum().reset_index()
df3 = recovered_df.groupby('Country/Region').sum().reset_index()

k =df1[df1['Country/Region']=='India'].loc[:,'2/4/20':]
india_confirmed = k.values.tolist()[0]

growth_diff = []

for i in range(1,len(india_confirmed)):
    growth_diff.append(india_confirmed[i] / india_confirmed[i-1])

growth_factor = sum(growth_diff)/len(growth_diff)
print('Average growth factor',growth_factor)
```

```
Average growth factor 1.1597143741171045
```

Prediction for the next 15 days

```
prediction_dates = []

start_date = dates_india[len(dates_india) - 1]
for i in range(15):
    date = start_date + datetime.timedelta(days=1)
    prediction_dates.append(date)
    start_date = date
previous_day_cases = global_confirmed[5][len(dates_india) - 1]
predicted_cases = []

for i in range(15):
    predicted_value = previous_day_cases *  growth_factor
    predicted_cases.append(predicted_value)
    previous_day_cases = predicted_value
```

Plotting the curve : -

```
plt.figure(figsize= (12,7))
plt.xticks(rotation = 90 ,fontsize = 11)
plt.yticks(fontsize = 10)
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Predicted Values for the next 15 Days" , fontsize = 20)
ax1 = plt.plot_date(y= predicted_cases,x= prediction_dates,linestyle ='-',color = 'r')
```

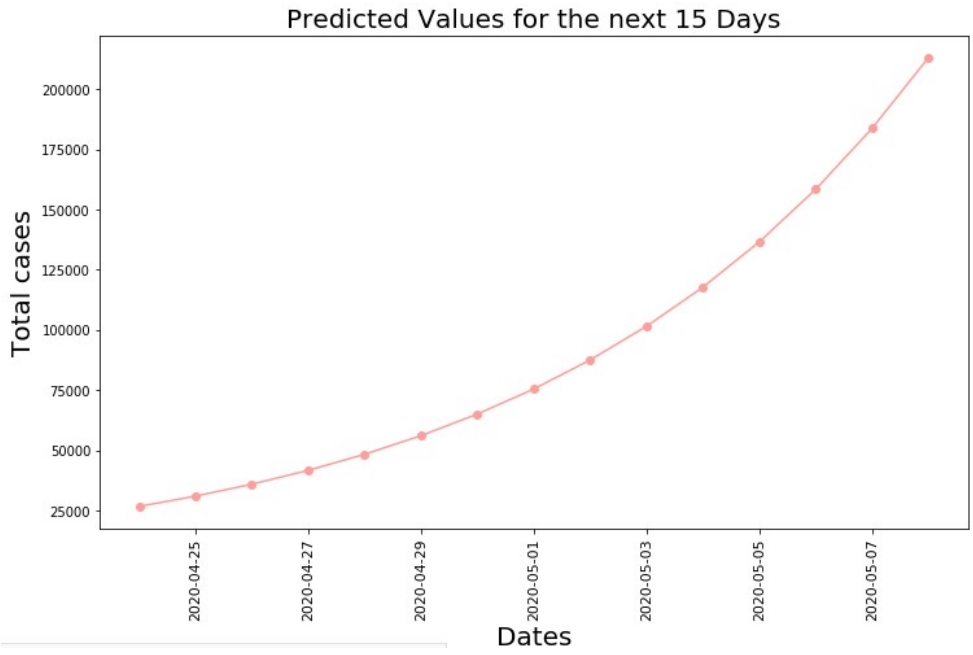

Fig. 22. The plot in the figure forecasts the growth in the number of cases in the next 15 days (starting May 01, 2020) given that the growth factor of the virus remains constant throughout the pandemic.

We could see that the graph is increasing exponentially if the average growth factor doesn't decrease.

It is important that the growth factor is reduced to flatten the curve.

4.4.2 Prediction using Prophet Model

We use the FB Prophet forecasting model to make predictions about the number of confirmed cases with respect to time. The seasonality feature of the prophet model is turned off because of the short duration of the data collection period.

```
k = df1[df1['Country/Region']=='India'].loc[:,'1/22/20':]
india_confirmed = k.values.tolist()[0]
data = pd.DataFrame(columns = ['ds','y'])
data['ds'] = dates
data['y'] = india_confirmed

prop=Prophet()
prop.fit(data)
future=prop.make_future_dataframe(periods=30)
prop_forecast=prop.predict(future)
forecast = prop_forecast[['ds','yhat']].tail(30)

fig = plot_plotly(prop, prop_forecast)
fig = prop.plot(prop_forecast,xlabel='Date',ylabel='Confirmed Cases')
```

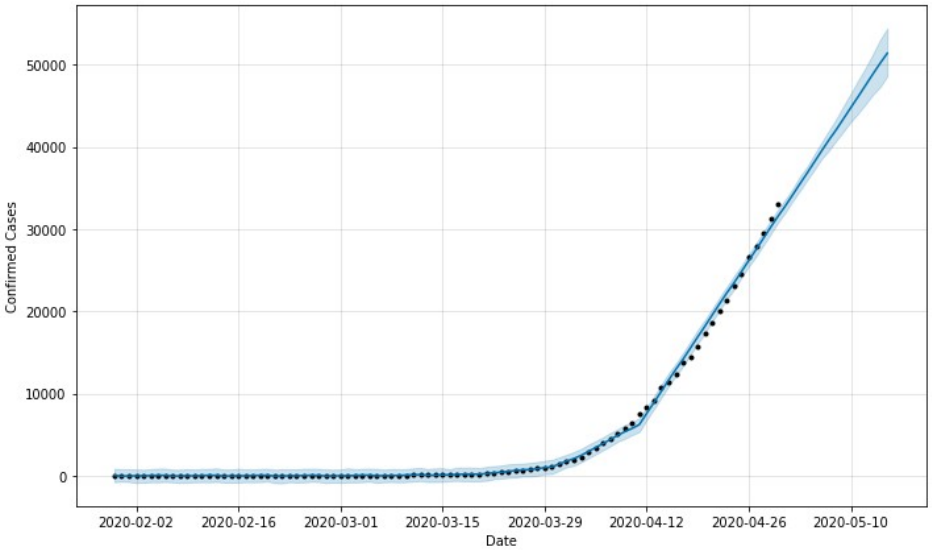

Fig. 23. In the figure, the black dots represent the actual data points; the blue line represents the predicted exponential growth in the number of confirmed COVID-19 cases, the shaded blue region represents the uncertainty in the forecast.

4.4.3 Prediction using ARIMA

Here, we use the ARIMA model to predict the rise of cases from 01-05-2020 to 15-05-2020 which, evidenced by the graph plotted below, fits very nicely with the actual data.

```
arima = ARIMA(data['y'], order=(5, 1, 0))
arima = arima.fit(trend='c', full_output=True, disp=True)
forecast = arima.forecast(steps= 30)
pred = list(forecast[0])

start_date = data['ds'].max()
prediction_dates = []
for i in range(30):
    date = start_date + datetime.timedelta(days=1)
    prediction_dates.append(date)
    start_date = date
plt.figure(figsize= (12,7))
plt.xlabel("Dates",fontsize = 20)
plt.ylabel('Total cases',fontsize = 20)
plt.title("Predicted Values for the next 15 Days" , fontsize = 20)

plt.plot_date(y= pred,x= prediction_dates,linestyle ='dashed',color = '#ff9999',label = 'Predicted');
plt.plot_date(y=data['y'],x=data['ds'],linestyle = '-',color = 'blue',label = 'Actual');
plt.legend();
```

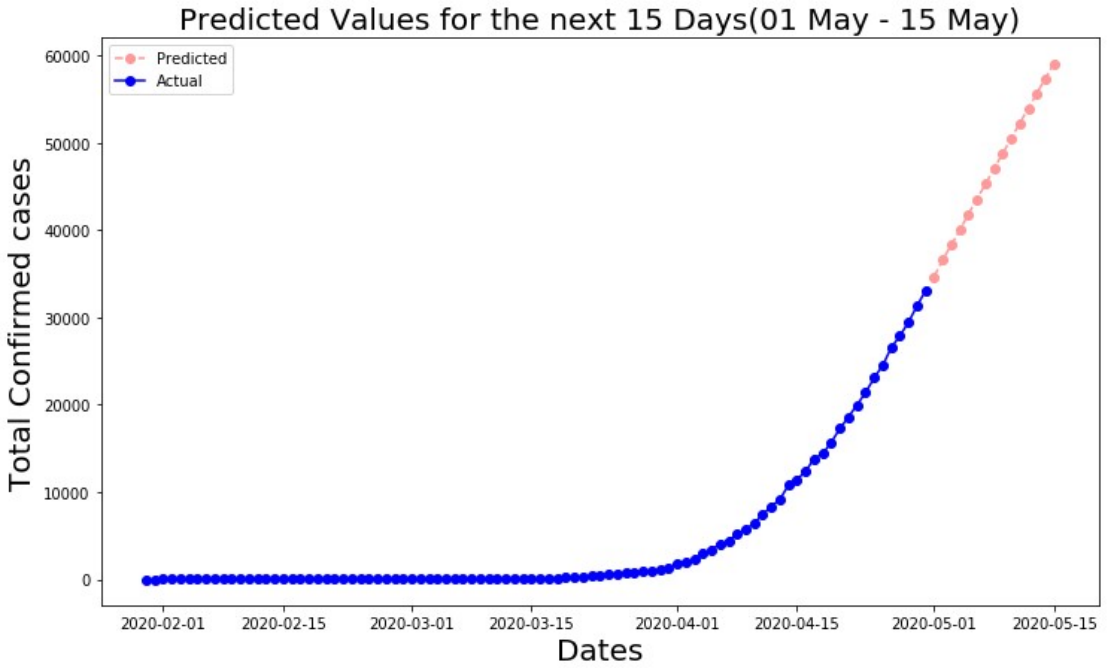

Fig. 24. In the figure the dotted blue curve represents the real data points of the COVID-19 cases in India, and the dotted orange curve represents the predicted number of cases for the next 15 days (starting from 01 May 2020).

4.4.4 LGBM Regressor

Implementation :-

```
lgbm = LGBMRegressor(n_estimators=1300)
lgbm.fit(x_train,y_train)
pred = lgbm.predict(x_test)
lgbm_forecast = lgbm.predict(test_india)
models = []
mse = []
mae = []
rmse = []
models.append('LGBM')
mse.append(round(mean_squared_error(pred, y_test),2))
mae.append(round(mean_absolute_error(pred, y_test),2))
rmse.append(round(np.sqrt(mean_squared_error(pred, y_test)),2))
```

4.4.5 Random Forest Regressor

Implementation :-

```
rf = RandomForestRegressor(n_estimators=100)
rf.fit(x_train,y_train)
pred = rf.predict(x_test)
rfr_forecast = rf.predict(test_india)
models.append('Random Forest')
mse.append(round(mean_squared_error(pred, y_test),2))
mae.append(round(mean_absolute_error(pred, y_test),2))
rmse.append(round(np.sqrt(mean_squared_error(pred, y_test)),2))
```

4.4.6 XGB Regressor

XGboost called for extreme Gradient Boosted trees. We have encountered a lot of tree-based algorithms like decision trees, in those we used to train our single model on a particular dataset maybe with some parameter tuning. Also in ensemble models, we used to train all the models separately.

Boosting is also an ensemble technique, which combines many models to give a final one but rather than evaluating all models separately, boosting trains models in sequence. that means, every new model is trained to correct the error of the previous model and the sequence got stopped when there is no further improvement. That is why it is more accurate.

Implementation :-

```
xgb = XGBRegressor(n_estimators=100)
xgb.fit(x_train,y_train)
pred = xgb.predict(x_test)
xgb_forecast = xgb.predict(test_india)
models.append('XGBoost')
mse.append(round(mean_squared_error(pred, y_test),2))
mae.append(round(mean_absolute_error(pred, y_test),2))
rmse.append(round(np.sqrt(mean_squared_error(pred, y_test)),2))
```

In all the models used above for making prediction, we have stored the **Mean Squared Error (MSE)**, **Mean Average Error (MAE)** and **Root Mean Squared Error (RMSE)** in separate lists. We will use these lists to plot a graph to understand which of our models makes best predictions.

4.4.7 Effects of Lockdown

In this section we study the change in growth factor under different lockdown scenarios. This will give us a good idea about the effectiveness of lockdowns in India.

- **Phase 1: Without lockdown (30 Jan - 24 Mar)**

The first case in India was recorded on 30th January 2020 in Kerala. From this date to 24th march was the starting of the pandemic in India which would get even worse as it progresses.

```
india_confirmed_without_ld = grouped_by_dates_without_ld['Total Confirmed cases']

growth_without_ld_diff = []

for i in range(1,len(india_confirmed_without_ld)):
    growth_without_ld_diff.append(india_confirmed_without_ld[i] / india_confirmed_without_ld[i-1])

growth_factor_without_ld = sum(growth_without_ld_diff)/len(growth_without_ld_diff)

growth.append(growth_factor_without_ld)
print('Average growth factor',growth_factor_without_ld)
```

```
Average growth factor 18.943995231637796
```

- **Phase 2: Lockdown 1.0 (25 Mar – 14 Apr)**

The first lockdown in India started on 25th March and lasted 21 days, till 14th April 2020. This was the first nationwide step taken by the India government to curb the effect of the virus in India.

```
india_confirmed_ld1 = grouped_by_dates_ld1['Total Confirmed cases']

growth_ld1_diff = []

for i in range(1,len(india_confirmed_ld1)):
    growth_ld1_diff.append(india_confirmed_ld1[i] / india_confirmed_ld1[i-1])

growth_factor_ld1 = sum(growth_ld1_diff)/len(growth_ld1_diff)

growth.append(growth_factor_ld1)
print('Average growth factor',growth_factor_ld1)
```

```
Average growth factor 1.0939853142698885
```

- **Phase 3: Lockdown 2.0 (15 Apr – 30 Apr*)**

The second lockdown in India started on 15th April 2020 and lasted 19 days till 3rd May 2020. But this study was performed on the data till 30th of April *.

```
india_confirmed_ld2 = grouped_by_dates_ld2['Total Confirmed cases']

growth_ld2_diff = []

for i in range(1,len(india_confirmed_ld2)):
    growth_ld2_diff.append(india_confirmed_ld2[i] / india_confirmed_ld2[i-1])

growth_factor_ld2 = sum(growth_ld2_diff)/len(growth_ld2_diff)

growth.append(growth_factor_ld2)
print('Average growth factor',growth_factor_ld2)
```

```
Average growth factor 1.0734726707398878
```

Chapter 5

Results and Discussion

The growth factor model fitting for the COVID-19 pandemic with respect to the daily reported cases in India is shown in figure 4.

After analyzing the data, the calculated growth factor turned out to be 1.5625 (which is less than the agreed-upon range of transmission rates i.e., between 2 and 3; source WHO). Therefore, figure 4 shows the exponential growth curve and the surge in the number of cases if the growth rate remains constant. This assumption of constant growth rate is not accurate to assume since several prevention and curing measures taken by the government tend to decrease the growth factor [17]. So, we use Prophet model for forecasting the number of cases because of its ability to incorporate complex behaviour of data, ease with time-series data and produce forecasts with high confidence levels. The yearly seasonality in the model is turned off because of obvious reasons.

Therefore, we use Prophet (see figure 23) forecasting model to predict the number of cases for the next 15 days i.e., from 01-05-2020 to 15-05-2020. Model forecast shows very low uncertainty in the prediction values as can be extrapolated by the plot above. The model predicts with 95% confidence that the number of confirmed cases in the country will surpass 50,000 by 12-05-2020.

ARIMA model is excellent for time-series non-stationary data. Our data has same characteristics therefore we use ARIMA to simulate the number of cases in the country for future dates. We have used the p value (refer section 4.4.3) of 5 for the autoregressive model and the degree of differencing, d (refer section 4.4.3) is set to 1. Therefore, ARIMA (5,1,0) model is used in our case. As shown in above figure, the actual data and the predicted curves seem to follow similar trends. We also used some decision tree machine learning regressor models to see how well they work on our data prediction. We used Random Forest Regressor, Light Gradient Boosting Model and XGBoost Regressor to test their performance. We use RMSE values to show the best fitted model (see figure 25) out of all three traditional machine learning models.

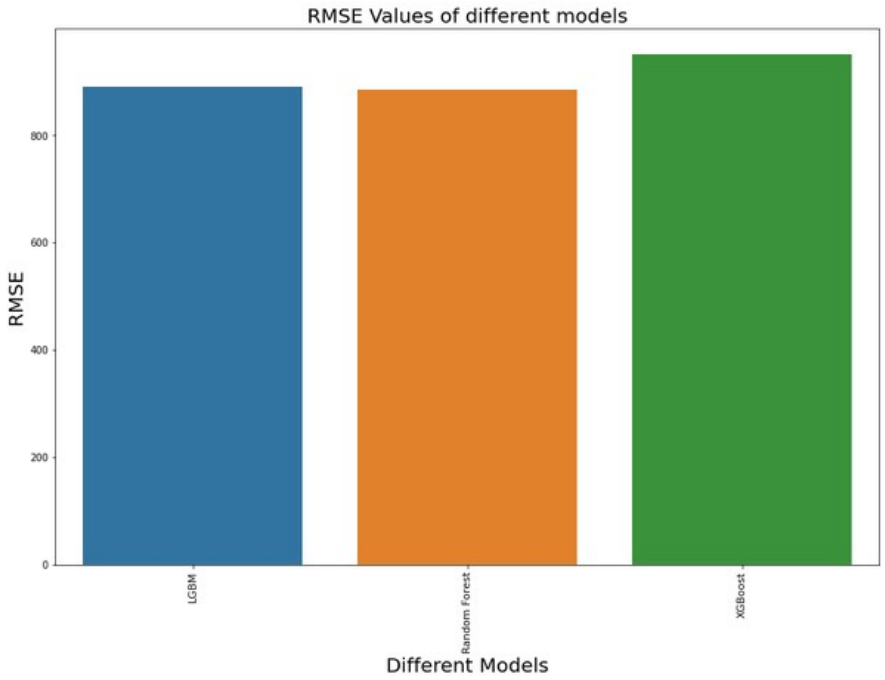

Fig.25. Root Mean Square Error for each Model

The regressors were fit on the training data which was chosen to be from 30-01-2020 (The first case recorded in India) to 12-04-2020. The regressors were tested on the rest of the data i.e., till 30-04-2020. The above figure gives the root mean square error values of all three models. We understand that random forest and light gradient boosting algorithms show similar results. Moreover, Xgboost algorithm gives higher error rate in prediction of the number of cases.

5.1 Effects of Social Distancing

To investigate the lock-down effect in India, we estimate a daily increase of cases in different stages of measures taken by the government of India. Information of growth factor for India under different lock-down scenarios is depicted in Table 1. In the initial days of the pandemic i.e., without the lockdown period the growth factor was very high as compared to other countries. With the massive population like that of India this would have been disastrous. But thankfully the government imposed strict police enforced lockdowns all across the country to curb the pandemic. Estimates of growth factors clearly show that it falls dramatically during the first lock-down period (25/03/2020 till 14/04/2020) and in the second lock-down period. The growth factors remain very close to unity during both the lock-down periods.

Table 1. The above table shows the growth factor in each stage of the pandemic in India. *The lockdown 2.0 lasted till 3rd May 2020 but this paper uses data till 30th April 2020.

Stage	Start Date	End Date	Growth Factor
Without Lockdown	30 Jan, 2020	24 March, 2020	18.943
Lockdown 1.0	25 March, 2020	14 April, 2020	1.093
Lockdown 2.0	15 April, 2020	30 April, 2020*	1.073

This indicates that the number of new notified cases will decrease after the lock-down periods are over in India. This result further supports the fact that both lockdowns were effective and should be extended further.

Chapter 6

Conclusion

In this project, the relevant models are chosen by examining the current data of the Indian pandemic situation, and then the simulation is performed. Moreover, we forecast the evolution of current situation and observed that the enforcement of controls would have a major effect on the outbreak. We have reviewed the situation of Covid-19 spread in India in realistic scenarios. Here, we studied the main impacts on the Indian society by the outbreak of COVID-19, such as its impact on the people at both national and district level. We also analyzed the countermeasures taken by the authorities to combat this pandemic and its success and failures throughout the country. The effect of lock-down has been discussed with different stages of lockdown. It should be noted that a sudden increase in infected people may give rise to a severe situation in terms of total cases, which can overwhelm the medical institutions The studies also show the effect of lock-down has been significant in dealing with Covid19. Infectious disease is a social concern in that it can inflict both personal and societal damage on a population. Therefore, numerous research in different fields are being carried out to minimize the social losses by predicting the spread of infectious diseases. The aim of this study was to analyze the data of the COVID-19 outbreak and produce some intelligent forecasts to understand the trend of increase in number of reported cases. The results of the analysis show that the number of cases will increase in the next 15 days.

Therefore it is imperative for the government to extend the lockdown period to reduce the strain on the medical organizations and help in flattening the curve. This research also has a drawback, which is a fairly short duration of data collection. India is at an elevated risk of entering community level transmission (Stage 3) due to failure in following the quarantine guidelines by citizens, as well as other social and demographic challenges [21]. The results and forecasting of the evolution of the pandemic made using the current data by our models in the current study will be invalid in such case. For the COVID-19 situation to end as soon as possible, it is essential for every individual to be responsible for their safety and others. Finally, the results that we get using our models solely depends on the data. Because of real-time modifications in data each and every day, the predictions could increase or decrease accordingly. Consequently, the results and predictions of this paper should only be used for qualitative interpretation and rational assessment of the complexity of the pandemic.

Finally, we hope that this study can make some contributions to the India's response to this pandemic and that we succeeded in making some useful insights for the policy makers of the country to use.

6.1 Future Scope and Research

The modeling and simulation of epidemics through a population is a complex process. Studies have shown that the infection caused by virus outbreaks closely resemble gaussian distribution [18, 22]. This mathematical tool can be used to construct an effective models. Deep learning methods like DNN (Deep Neural Networks) and LSTM (Long Short Term Memory) methods can prove very effective in capturing the intricate behaviour of the pandemics like COVID-19 [19].

References

1. "Coronavirus Disease 2019" , https://www.who.int , Retrieved: [30 April, 2020].
2. Mattia Mori, Clemente Capasso, Fabrizio Carta, William a Donald &Claudiu T Supuran (2020). "A deadly spillover: SARS-CoV-2 outbreak", Expert Opinion on Therapeutic Patents, DOI: 10.1080/13543776.2020.1760838
3. "COVID-19 Coronavirus Pandemic" ,https://www.worldometers.info/coronavirus/ , Retrieved : [30 April, 2020].
4. David Reid, Jan 30, 2020, "Coronavirus (COVID-19)" , https://www.cnbc.com, Retrieved : [01 May, 2020]
5. "COVID-19" , AVAILABLE: https://www.mohfw.gov.in/, Retrieved : [30-04-2020]
6. Vaishnavi Chandrashekhar, 30 March, 2020, "Coronavirus Disease 2019" , https://www.sciencemag.org/ , Retrieved : [01-04-2020].
7. Zhai, Pan, et al. "The Epidemiology, Diagnosis and Treatment of COVID-19." International Journal of Antimicrobial Agents, 2020, p. 105955.
8. Hongjun Zhu, "Transmission Dynamics and Control Methodology of COVID-19: a Modeling Study" , medRxiv , 2020.03.29.20047118
9. Taylor SJ, Letham B. 2017. "Forecasting at scale". PeerJ Preprints 5:e3190v2 https://doi.org/10.7287/peerj.preprints.3190v2
10. Steven Liu, Dec 21, 2018, "Forecasting with Prophet". [On-line]. Available: https://towardsdatascience.com . Retrieved: [Apr 30, 2020].
11. "ARIMA models for time series forecasting", Available: https://people.duke.edu/~rnau/411arim.htm . Retrieved: [Apr 30, 2020].
12. Krishni , Nov 27, 2018, "Guide to Random Forest Regression", [On-line]. Available: https://medium.com , Retrieved: [May 01, 2020].
13. Jason Brownlee , August 21, 2019 , "A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning", Available: https://machinelearningmastery.com Retrieved : [April 30, 2020].
14. Chen, Tianqi & Guestrin, Carlos. (2016). "XGBoost: A Scalable Tree Boosting System". 785-794. 10.1145/2939672.2939785.
15. "COVID-19 Stats", Available: https://www.covid19india.org/ .
16. Nick Triggle ,16 April,2020, "Coronavirus: How to understand the death tol"l , Available: https://www.bbc.com. Retrieved : [April 30,2020].
17. Samantha Sault , 21 March, 2020," Why lockdowns can halt the spread of COVID-19", Available: https://www.weforum.org .Retrieved : [May 2, 2020].
18. Lixiang Li, Zihang Yang, Zhong kai Dang, Cui Meng, Jingze Huang, Haotian Meng, Deyu Wang, Guanhua Chen, Jiaxuan Zhang, Haipeng Peng, Yiming Shao , "Propagation analysis and prediction of the COVID-19" , Infectious Disease Modelling, Volume 5 , 2020, Pages 282-292.

19. Chae, Sangwon et al. “Predicting Infectious Disease Using Deep Learning and Big Data.” International journal of environmental research and public health vol. 15,8 1596. 27 Jul. 2018, doi:10.3390/ijerph15081596.
20. Kirti Pandey, 01 May, 2020, “Lockdown Timeline”, Available: https://www.timesnownews.com Retrieved: [May 02, 2020].
21. Amit Mudgill, 25 March, 2020, “How will lockdown play out for India”, Available: https://economictimes.indiatimes.com Retrieved: [May 02, 2020].
22. Predictions for COVID-19 Outbreak In India Using Epidemiological Models, Rajesh Ranjan, doi: https://doi.org/10.1101/2020.04.02.20051466
23. M.K., Arti. (2020). Modeling and Predictions for COVID 19 Spread in India. 10.13140/RG.2.2.11427.81444.