Data Mining and Analysis Assignment

on

COVID-19 Trends and Prediction

A Project Report submitted in partial fulfilment of the

requirements for the award of

BACHELOR OF ENGINEERING

IN COMPUTER SCIENCE AND ENGINEERING

Submitted By

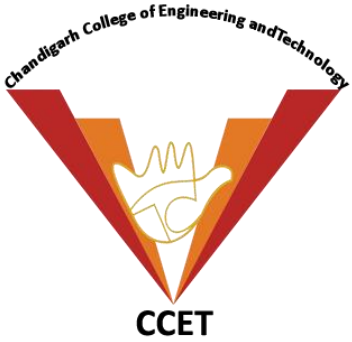
Aarushi Sood (CO17301)

Keshav Tangri (CO17333)

Under the supervision of

Dr. Ankit Gupta, Assistant Professor,

CSE Dept. CCET (Degree Wing)



CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY

(DEGREE WING)

Government Institute under Chandigarh (UT) Administration, Affiliated to Panjab University, Chandigarh

Sector – 26, Chandigarh PIN - 160019

April, 2020

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)

Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh

Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943

Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Sc.& Engineering

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled "COVID-19 Trends and Prediction", in fulfilment of the requirement for the award of the degree Bachelor of Engineering in Computer Science & Engineering, submitted in CSE Department, Chandigarh College of Engineering & Technology(Degree wing) affiliated to Punjab University, Chandigarh, is an authentic record of my/our own work carried out during my degree under the guidance of Assistant Professor. Dr. Ankit Gupta of Computer Science Engineering, CCET, Sector 26, Chandigarh. The work reported in this has not been submitted by me for award of any other degree or diploma.

Date : Aarushi Sood(CO17301)

Place : CCET, Sector 26, Chandigarh Keshav Tangri(CO17333)

Department of Computer Sc.& Engineering

ACKNOWLEDGEMENT

*I would like to place on record my deep sense of gratitude to **Dr. Ankit Gupta, Assistant Professor, Dept. of Computer Science & Engineering, CCET(Degree), Chandigarh** for their generous guidance, help and useful suggestions.*

*I express my sincere gratitude to my **parents** who patiently helped me as I went through my work and helped to modify and eliminate some of the irrelevant or unnecessary stuff.*

*I would also like to thank my **friends** who helped me to make my work more organized and well-stacked till the end.*

*Last but not the least, I would thank The **Almighty** for giving me strength to complete my report on time.*

Aarushi Sood (CO17301)

Keshav Tangri (CO17333)

CSE 3rd Year

Department of Computer Sc.& Engineering

ABSTRACT

Coronavirus disease (COVID-19) is an infectious disease caused by a newly discovered coronavirus.

Most people infected with the COVID-19 virus will experience mild to moderate respiratory illness and recover without requiring special treatment. Older people, and those with underlying medical problems like cardiovascular disease, diabetes, chronic respiratory disease, and cancer are more likely to develop serious illness.

The best way to prevent and slow down transmission is to be well informed about the COVID-19 virus, the disease it causes and how it spreads. Protect yourself and others from infection by washing your hands or using an alcohol-based rub frequently and not touching your face.

In order to study and analyse, a model is implemented that provides an interactive and visually pleasing plots, prediction analysis on the state level and also the country wise confirmed plot. The major **benefits** of the model developed are :

- It provides state level predictions for the increase in number of cases at each day.
- State wise analysis of hospitals and categorization.
- Maintaining a count on total positive and negative corona cases, at each state, hence maintaining test labs and details for each state.
- Extending the analysis and visualization to the data of various other countries

CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 ABOUT CORONA VIRUS

Government of India is taking all necessary steps to ensure that we are prepared well to face the challenge and threat posed by the growing pandemic of COVID-19 the Corona Virus. With active support of the people of India, we have been able to contain the spread of the Virus in our country. The most important factor in preventing the spread of the Virus locally is to empower the citizens with the right information and taking precautions as per the advisories being issued by Ministry of Health & Family Welfare.

Preventive Measures Against Corona Virus:

To prevent infection and to slow transmission of COVID-19, do the following:

- Wash your hands regularly with soap and water, or clean them with alcohol-based hand rub.
- Maintain at least 1 metre distance between you and people coughing or sneezing.
- Avoid touching your face.
- Cover your mouth and nose when coughing or sneezing.
- Stay home if you feel unwell.
- Refrain from smoking and other activities that weaken the lungs.
- Practice physical distancing by avoiding unnecessary travel and staying away from large groups of people.

Symptoms of Corona Virus:

The COVID-19 virus affects different people in different ways. COVID-19 is a respiratory disease and most infected people will develop mild to moderate symptoms and recover without requiring special treatment. People who have underlying medical conditions and those over 60 years old have a higher risk of developing severe disease and death.

Common symptoms include:

- fever
- tiredness

- dry cough.

Other symptoms include:

- shortness of breath
- aches and pains
- sore throat
- and very few people will report diarrhoea, nausea or a runny nose.

People with mild symptoms who are otherwise healthy should self-isolate and contact their medical provider or a COVID-19 information line for advice on testing and referral.

People with fever, cough or difficulty breathing should call their doctor and seek medical attention.

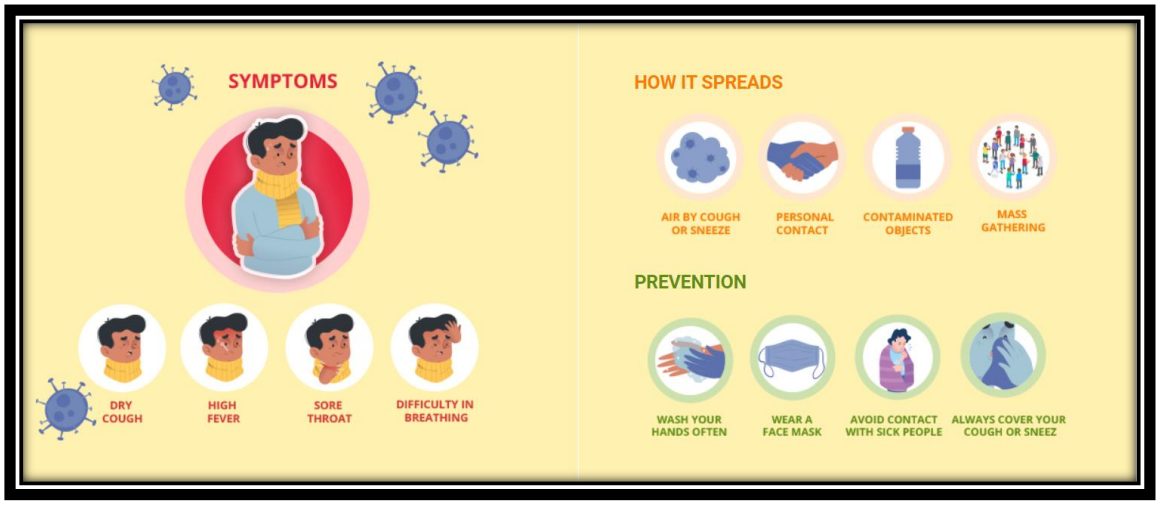


Figure 1 : Coronavirus symptoms, spread and prevention

1.2 ABOUT MACHINE LEARNING

Machine learning is a method of data analysis that automates analytical model building. It is a branch of artificial intelligence based on the idea that systems can learn from data, identify patterns and make decisions with minimal human intervention.

It's Importance :

Resurging interest in machine learning is due to the same factors that have made data mining and Bayesian analysis more popular than ever. Things like growing volumes and varieties of available data, computational processing that is cheaper and more powerful, and affordable data storage.

All of these things mean it's possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, more accurate results – even on a very large scale. And by building precise models, an organization has a better chance of identifying profitable opportunities – or avoiding unknown risks.

Machine Learning Methods :

- **Supervised learning** algorithms are trained using labelled examples, such as an input where the desired output is known. For example, a piece of equipment could have data points labelled either “F” (failed) or “R” (runs). The learning algorithm receives a set of inputs along with the corresponding correct outputs, and the algorithm learns by comparing its actual output with correct outputs to find errors. It then modifies the model accordingly. Through methods like classification, regression, prediction and gradient boosting, supervised learning uses patterns to predict the values of the label on additional unlabelled data. Supervised learning is commonly used in applications where historical data predicts likely future events. For example, it can anticipate when credit card transactions are likely to be fraudulent or which insurance customer is likely to file a claim.

- **Unsupervised learning** is used against data that has no historical labels. The system is not told the "right answer." The algorithm must figure out what is being shown. The goal is to explore the data and find some structure within. Unsupervised learning works well on transactional data. For example, it can identify segments of customers with similar attributes who can then be treated similarly in marketing campaigns. Or it can find the main attributes that separate customer segments from each other. Popular techniques include self-organizing maps, nearest-neighbour mapping, k-means clustering and singular value decomposition. These algorithms are also used to segment text topics, recommend items and identify data outliers.

- **Semisupervised learning** is used for the same applications as supervised learning. But it uses both labelled and unlabelled data for training – typically a small amount of labelled data with a large amount of unlabelled data (because unlabelled data is less expensive and takes less effort to acquire). This type of learning can be used with methods such as classification, regression and prediction. Semisupervised learning is useful when the cost associated with labelling is too high to allow for a fully labelled training process. Early examples of this include identifying a person's face on a web cam.

- **Reinforcement learning** is often used for robotics, gaming and navigation. With reinforcement learning, the algorithm discovers through trial and error which actions yield the greatest rewards. This type of learning has three primary components: the agent (the learner or decision maker), the environment (everything the agent interacts with) and actions (what the agent can do). The objective is for the agent to choose actions that maximize the expected reward over a given amount of time. The agent will reach the goal much faster by following a good policy. So the goal in reinforcement learning is to learn the best policy.

1.3 ABOUT THE MODEL IMPLEMENTED

The model implemented will segregate the data state wise and for each state, it will plot the line chart for no. of confirmed cases on each day along with the cured and deaths lines on the same plot. Next, per day increase on Cases will be taken into consideration and plotted against dates. This is found to be a skewed graph and this skewness is handled by 2 approaches, namely the Logarithm approach and Cube root approach. After this the outliers in both the cases are compared. The graph having per day increase against date is used for model training and then after the model is trained and tested, the prediction for current date is given to the user. This is repeated for all the state and union territories. Final table is printed on the screen with predicted values for different regions. The in depth explanation of the Model is given in chapter 5.

CHAPTER 2

TECHNOLOGY STACK

Programming Language:

- Python

Graphing Library:

- Seaborn
- Plotly

Mathematical/Numerical Library:

- Numpy

Data Frame Library:

- Pandas

Machine Learning Library:

- Sklearn

2.1 Programming Language:

- **Python:** Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together. Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

2.2 Graphing Library:

- **Seaborn:** Seaborn is a Python library created for enhanced data visualization. It's a very timely and relevant tool for data professionals working today precisely because effective data visualization – and communication in general – is a particularly essential skill. Being able to bridge the gap between data and insight is hugely valuable, and

Seaborn is a tool that fits comfortably in the toolchain of anyone interested in doing just that.

There are, of course, a huge range of data visualization libraries out there – but if you're wondering why you should use Seaborn, put simply it brings some serious power to the table that other tools can't quite match.

- **Plotly:** Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.
  Plotly.py is free and open source and you can view the source, report issues or contribute on GitHub.

2.3 Mathematical/Numerical Library:

- **Numpy:** NumPy is the fundamental package for scientific computing in Python. It is a Python library that provides a multidimensional array object, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

  At the core of the NumPy package, is the *ndarray* object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance.

2.4 Data Frame Library:

- **Pandas:** is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

2.5 Machine Learning Library:

- **Sklearn:** scikit-learn is a Python module for machine learning built on top of SciPy and is distributed under the 3-Clause BSD license.

CHAPTER 3

DESCRIBING PLOTS

Before beginning our discussion, it is mandatory to import the following libraries into your python environment for the smooth functioning of the code and model.

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import datetime as dt
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.ensemble import RandomForestRegressor
import warnings
import math
```

Figure 2: Importing Libraries

Matplotlib.pyplot:

Matplotlib.pyplot is a plotting library used for 2D graphics in python programming language. It can be used in python scripts, shell, web application servers and other graphical user interface toolkits.

Seaborn:

Seaborn is a Python library created for enhanced data visualization. It's a very timely and relevant tool for data professionals working today precisely because effective data visualization and communication in general is a particularly essential skill.

Types of Plots:

3.1 Pie Chart

Matplotlib supports pie charts using the pie() function. The matplotlib module can be used to create all kinds of plots and charts with Python.

A pie chart is one of the charts it can create, but it is one of the many.First import plt from the matplotlib module with the line *import matplotlib.pyplot as plt* Then use the method *plt.pie()* to create a plot.

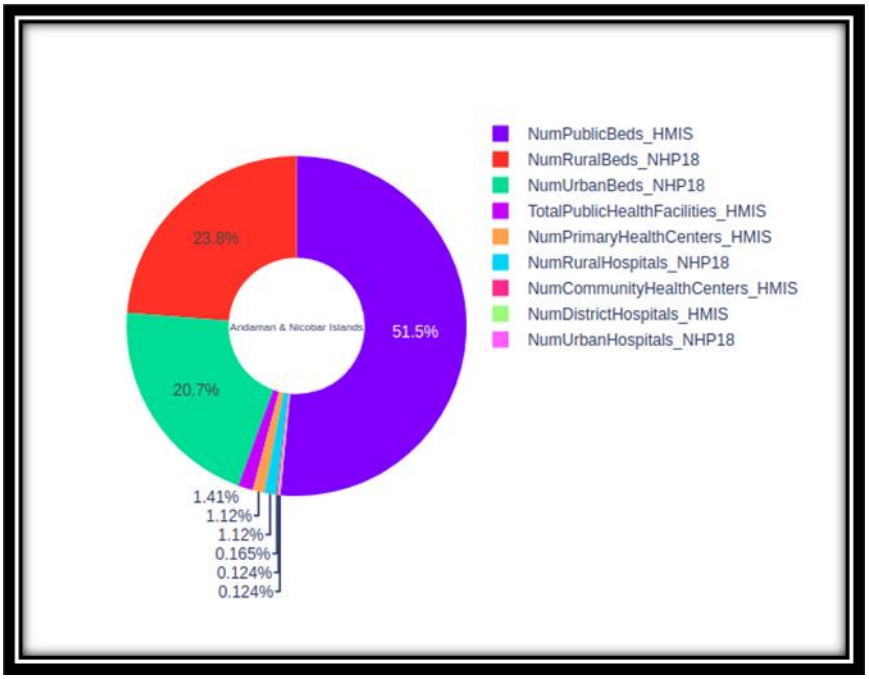


Figure 3: Example of Pie chart from our analysis

3.2 Bar Chart/Graph

A bar graph uses bars to compare data among different categories. It is well suited when you want to measure the changes over a period of time. It can be represented horizontally or vertically. Also, the important thing to keep in mind is that longer the bar, greater is the value.

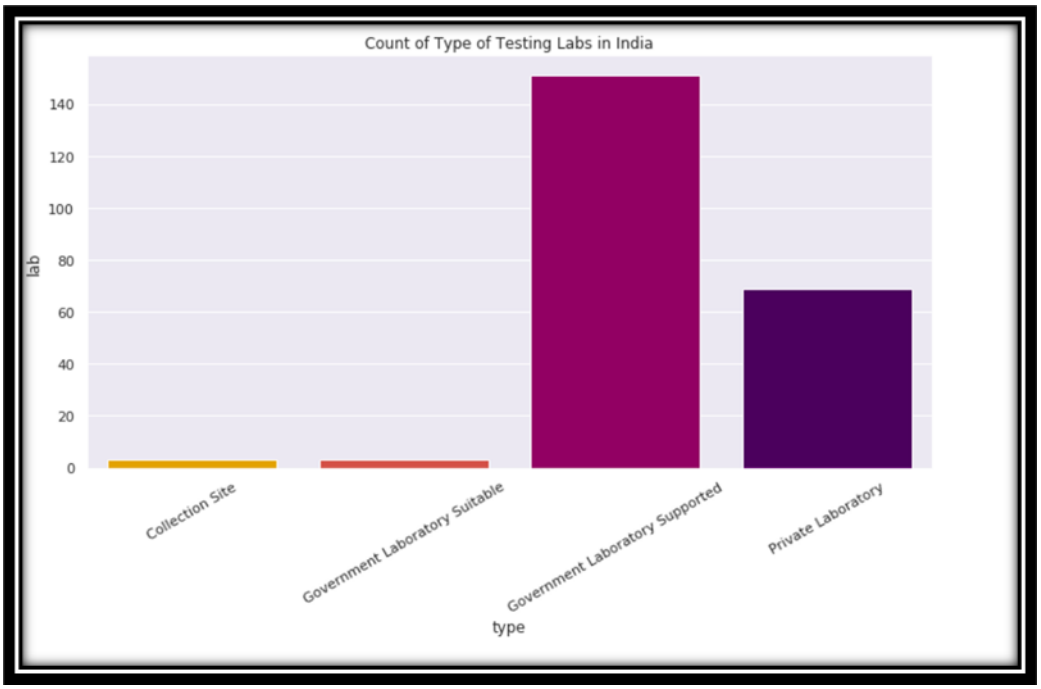


Figure 4: Example of Bar plot from our analysis

3.3 Box Plot

Boxplot is probably one of the most common type of graphic. It gives a nice **summary** of one or several **numeric variables**. The line that divides the box into 2 parts represents the **median** of the data.

The end of the box shows the upper and lower **quartiles**. The extreme lines shows the highest and lowest value excluding **outliers**. Note that boxplot hide the number of values existing behind the variable.

Thus, it is highly advised to print the number of observation, add unique observation with jitter or use a violinplot if you have many observations.

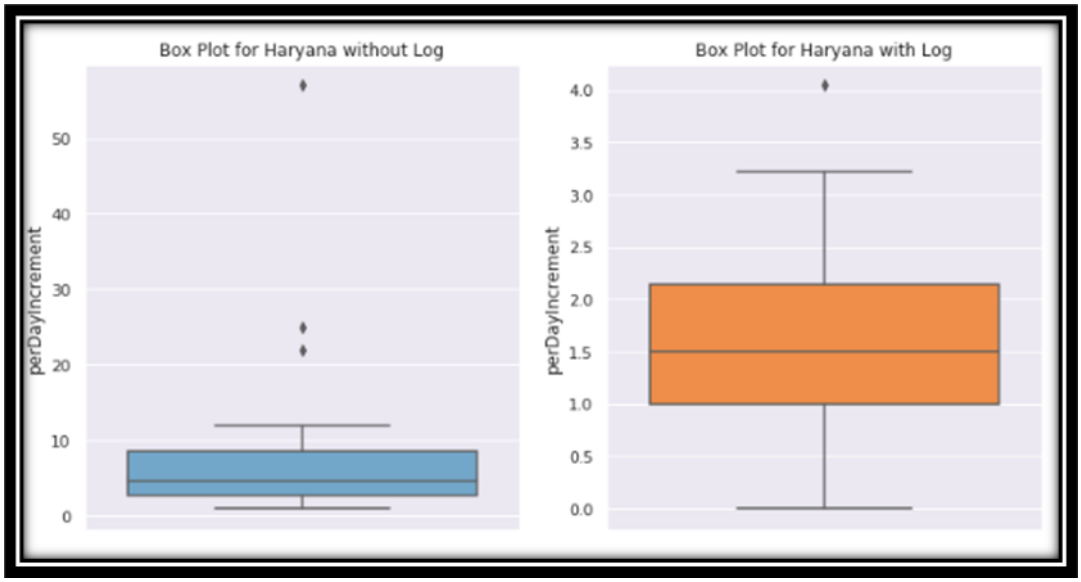


Figure 5: Example of Boxplots

3.4 Line Chart

A **line chart** or **line** graph is a type of **chart** which displays information as a series of data points called 'markers' connected by straight **line** segments. Create a new file, say it **line**.py and import matplotlib library in it. import matplotlib.pyplot as plt. The alias plt has been set for simplification purpose.

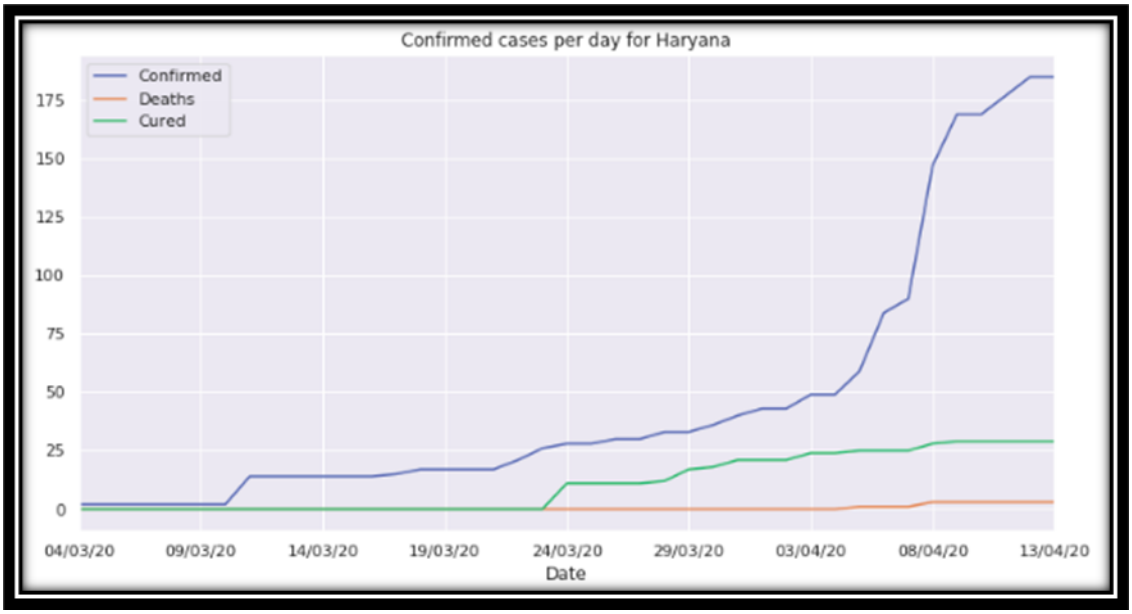


Figure 6 : Example of a Line Chart

3.5 Scatter Plot

Usually we need scatter plots in order to compare variables, for example, how much one variable is affected by another variable to build a relation out of it. The data is displayed as a collection of points, each having the value of one variable which determines the position on the horizontal axis and the value of other variable determines the position on the vertical axis.

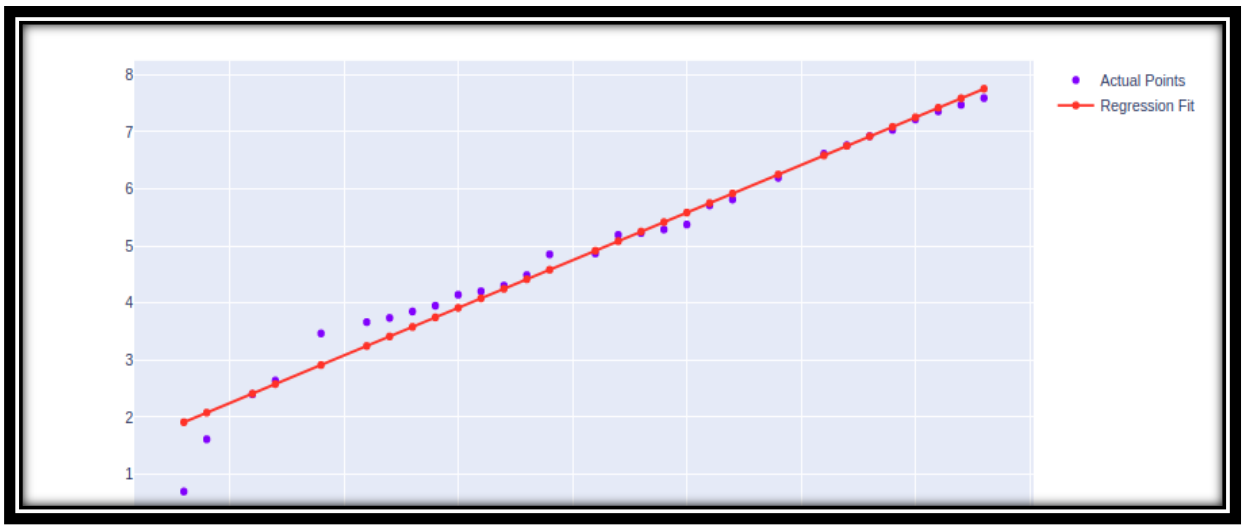


Figure 7: Example of Scatter plot and Line plot

CHAPTER 4 : ANALYSING THE DATA

After looking at the basic information in Introduction Chapter, getting to know about the Technology stack in chapter 2 , studying and understanding different Plot types in chapter 3 , we move onto understanding the Data we took under Consideration in this Assignment. Various csv files were considered to extract as much information possible about corona virus. The reference to all the data files will be given in the References page.

Following are the csv files Studied and Analysed during the course of this project :

1. time_series_covid_19_confirmed.csv

2. covid_19_india.csv

3. HospitalBedsIndia.csv

4. ICMRTestingLabs.csv

5. ICMRTestingDetails.csv

6. StatewiseTestingDetails.csv

**All the data files are being submitted along with this assignment **

1. time_series_covid_19_confirmed.csv

Starting with the first data set, This deals with the time series data for the spread of coronavirus, for all the countries. The data is extracted from kaggle and main source of the data is from the John Hopkins University. The time series data collection is started from 22nd january, 2020. The information about this file is shown below using pandas Library in Python

```
worldConfirmed = pd.read_csv('time_series_covid_19_confirmed.csv')
worldConfirmed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 263 entries, 0 to 262
Data columns (total 81 columns):
Province/State    82 non-null object
Country/Region    263 non-null object
Lat               263 non-null float64
Long              263 non-null float64
1/22/20           263 non-null int64
1/23/20           263 non-null int64
1/24/20           263 non-null int64
1/25/20           263 non-null int64
1/26/20           263 non-null int64
1/27/20           263 non-null int64
1/28/20           263 non-null int64
```

Figure 8

This file consists of 80+ Columns with the first column as Province/State of a Country , second column is the Corresponding Country , followed by the Latitude and Longitude of the State/Province/Country(if state/province is absent) . After that , starting from 22/01/2020 till the present date. The data is uploaded daily and each column in this time series data consists of Number of Confirmed cases in that country. This data helps us to analyse the growth curve of each country with respect to date to which the data has been recorded.

	Province/State	Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25
0	NaN	Afghanistan	33.00000	65.000000	0	0	0	
1	NaN	Albania	41.15330	20.168300	0	0	0	
2	NaN	Algeria	28.03390	1.659600	0	0	0	
3	NaN	Andorra	42.50630	1.521800	0	0	0	
4	NaN	Angola	-11.20270	17.873900	0	0	0	
...	...	...	...	...	...	...	...	
258	Falkland Islands (Malvinas)	United Kingdom	-51.79630	-59.523600	0	0	0	
259	Saint Pierre and Miquelon	France	46.88520	-56.315900	0	0	0	
260	NaN	South Sudan	6.87700	31.307000	0	0	0	
261	NaN	Western Sahara	24.21550	-12.885800	0	0	0	
262	NaN	Sao Tome and Principe	0.18636	6.613081	0	0	0	

263 rows × 81 columns

Figure 9

The above Image shows the contents of the csv file. As visible, the data doesn't consist for state/province for each Country/Region. As shown below, only the first Column has these NaN values.

```
worldConfirmed.isnull().sum()

Province/State    181
Country/Region      0
Lat                 0
Long                0
1/22/20             0
                 ...
4/3/20              0
4/4/20              0
4/5/20              0
4/6/20              0
4/7/20              0
Length: 81, dtype: int64
```

Figure 10

So, to clean this data, that is to incorporate this data , one option is to remove all the null rows, but that will lead to removal of 181 rows from the complete data of 263 rows. Thus decreasing the country count. So to tackle this , we can group the data frame by Country/Region, and then sum up the confirmed cases in their respective states for a particular date. This will lead to a new Data frame in which we won't have any NaN values and each date column will give the confirmed case count in respective Country/Region from 22nd january 2020 till date.

The grouping of the data and the resultant Data frame is shown below :

```
worldSeparate = worldConfirmed.groupby('Country/Region').aggregate(sum)
worldSeparate
```

Country/Region	Lat	Long	1/22/20	1/23/20	1/24/20	1/25/20	1/26/20	1/27/20	1/28/20
Afghanistan	33.0000	65.0000	0	0	0	0	0	0	0
Albania	41.1533	20.1683	0	0	0	0	0	0	0
Algeria	28.0339	1.6596	0	0	0	0	0	0	0
Andorra	42.5063	1.5218	0	0	0	0	0	0	0
Angola	-11.2027	17.8739	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...
Vietnam	16.0000	108.0000	0	2	2	2	2	2	2
West Bank and Gaza	31.9522	35.2332	0	0	0	0	0	0	0
Western Sahara	24.2155	-12.8858	0	0	0	0	0	0	0
Zambia	-15.4167	28.2833	0	0	0	0	0	0	0
Zimbabwe	-20.0000	30.0000	0	0	0	0	0	0	0

184 rows × 79 columns

Figure 11

Using this Grouped Dataframe, we will plot the everyday confimed cases of different countries to get insight about the Growth Rate of the respective Countries. The graph is ploted using the Plotly library for python. The library is explained in the Technology stack section. We will plot the growth curve for all the countries and get the result as :

```
fig = go.Figure()
for i in worldSeparate.index:
    fig.add_trace(go.Scatter(x=worldSeparate.columns[2:], y=worldSeparate.ix[i,2:] ,
                  mode='lines+markers',
                  name=i))
else:
    fig.show()
```

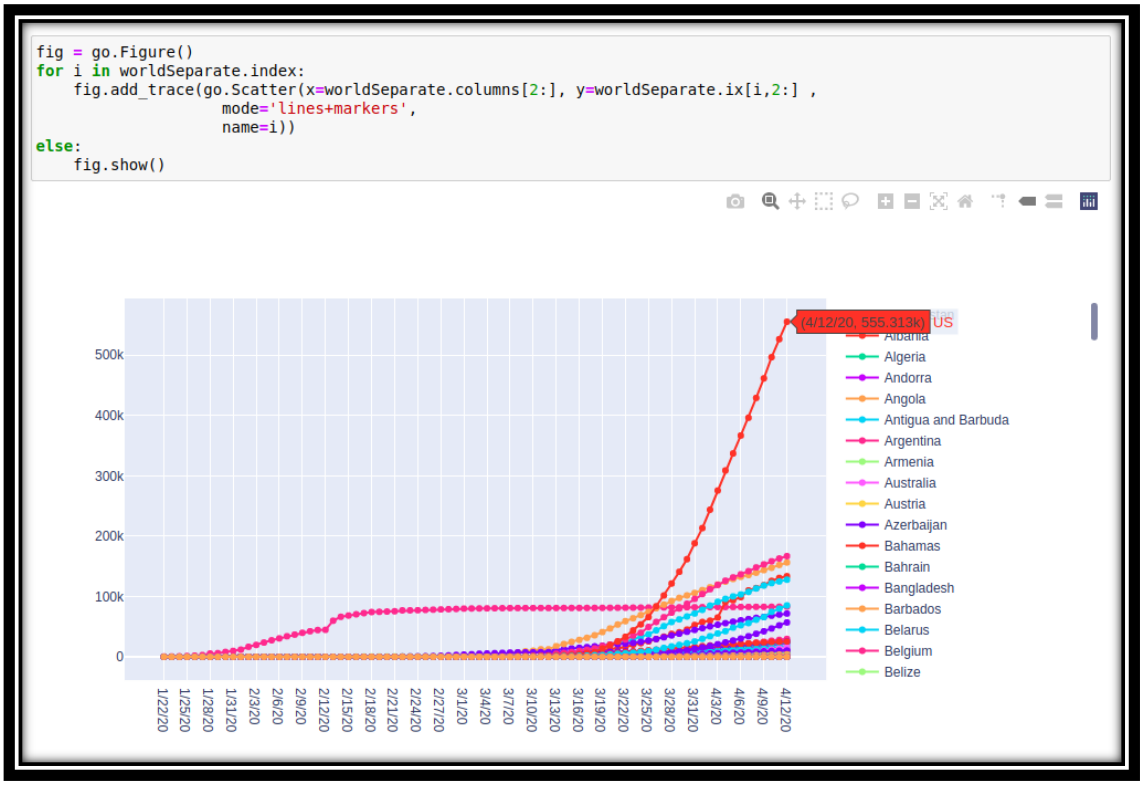


Figure 12

As visible from the Image , it can be seen that US has the Highest Growth rate in the number of confirmed corona positive cases with the number surpassing 550K cases as of 12th April 2020.

The growth curve of china has flattened at around 83K confirmed corona cases as of 12th April 2020.

United kingdom has surpassed china with 85K Confirmed corona positive cases as of 12th April 2020.

Spain has crossed Italy with 166K Confirmed Corona positive cases whereas Italy have 156K corona positive cases as of 12th April 2020.

2. covid_19_india.csv

This Data file deals with the growth of Coronavirus cases in India and is updated every day with the correct information from the ministry of health and family welfare website. First we will read the data file and see how the content is distributed.

```
data = pd.read_csv('covid_19_india.csv',index_col=0)
data.head()
```

Sno	Date	Time	State/UnionTerritory	ConfirmedIndianNational	ConfirmedForeignNational	Cured	Deaths	Confirmed
1	30/01/20	6:00 PM	Kerala	1	0	0	0	1
2	31/01/20	6:00 PM	Kerala	1	0	0	0	1
3	01/02/20	6:00 PM	Kerala	2	0	0	0	2
4	02/02/20	6:00 PM	Kerala	3	0	0	0	3
5	03/02/20	6:00 PM	Kerala	3	0	0	0	3

Figure 13

On analysing it was found that the ConfirmedIndianNational and ConfirmedForeignNational columns have a lot of '-' filled values along with Integer values. However we are not concerned about the unavailable data and this data can also be dropped as we only need to study Confirmed cases wrt Date. Along with this, we will drop the Time by using the following command.

```
data = data.drop(['ConfirmedIndianNational','ConfirmedForeignNational'
                  ,'Time'],axis = 1)
data.tail()
```

Sno	Date	State/UnionTerritory	Cured	Deaths	Confirmed
890	12/04/20	Telengana	43	9	504
891	12/04/20	Tripura	0	0	2
892	12/04/20	Uttarakhand	5	0	35
893	12/04/20	Uttar Pradesh	45	5	452
894	12/04/20	West Bengal	19	5	134

Figure 14

Next we will work on the state wise statistics for India. From the above data , we can extract information for Confirmed cured and Deaths in different states. For this we use the Command shown below :

```
stateWise = data.groupby(['State/UnionTerritory'])[['Confirmed','Cured','Deaths']].aggregate(max)
stateWise[['Confirmed','Cured','Deaths']].plot(kind = 'barh',figsize=(10,15))
```

Figure 15

Here we will group the data with respect to the State/UnionTerritory Column and for each respective state, we will take the max count of confirmed cases , Deaths and Cured cases. Then we will plot the details on a horizontal bar chart as shown below :

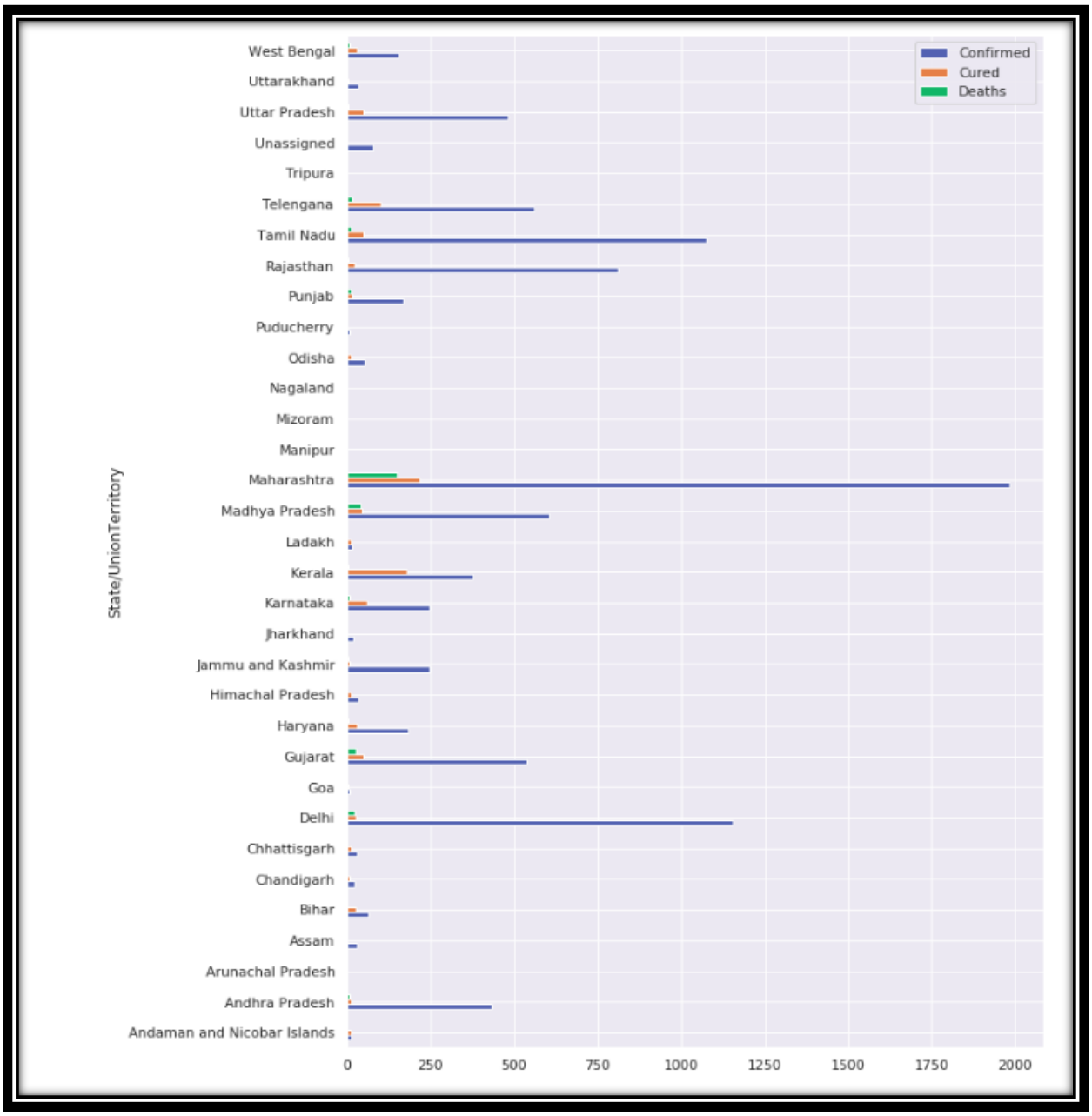


Figure 16

This shows that Maharashtra has the highest number of confirmed cases , followed by Delhi and then Tamil Nadu.

Mizoram , Arunachal Pradesh and Tripura have the least count of Confirmed cases and the rest of the states have varying number of cases.

The next phase of this data set analysis includes the state wise analysis about the growth of cases with respect to days passed. This part will be explained in general for One State and at the end of chapter 5, you will find all the data for all the states in India .

Note that, the data has been cleaned and has no missing values.

(i) Analysing the Confirmed Cases per day

Here we will extract the data for a particular state and then group this data in ascending order of the Date and confirmed cases. The extraction of data for a particular State and its first 5 entries and last 5 entries are shown as below. For the purpose of understanding, we will consider the example of the state : Maharashtra.

```
stateData = data[
            (data['State/UnionTerritory'] == "Maharashtra")
          ]
stateData.head()#First 5 entries
```

Sno	Date	State/UnionTerritory	Cured	Deaths	Confirmed
77	09/03/20	Maharashtra	0	0	2
92	10/03/20	Maharashtra	0	0	5
98	11/03/20	Maharashtra	0	0	2
121	12/03/20	Maharashtra	0	0	11
134	13/03/20	Maharashtra	0	0	14

```
stateData.tail()#last 5 entries
```

Sno	Date	State/UnionTerritory	Cured	Deaths	Confirmed
789	09/04/20	Maharashtra	117	72	1135
820	10/04/20	Maharashtra	125	97	1364
851	11/04/20	Maharashtra	188	110	1574
882	12/04/20	Maharashtra	208	127	1761
913	13/04/20	Maharashtra	217	149	1985

Figure 17

We will study the increase in the confirmed cases for each day for this data. Here , we will group the data with respect to the Date in ascending order, and then using plot function in the

pandas data frame to plot the confirmed cases, the cured cases and the deaths along the y axis. The code implementation of the same and its out put is given below :

```
stateData[['Date','Confirmed','Deaths','Cured']].set_index('Date')\
.plot(kind = 'line',title = "Confirmed cases per day for "+"Maharashtra
      figsize=(12,8));
```

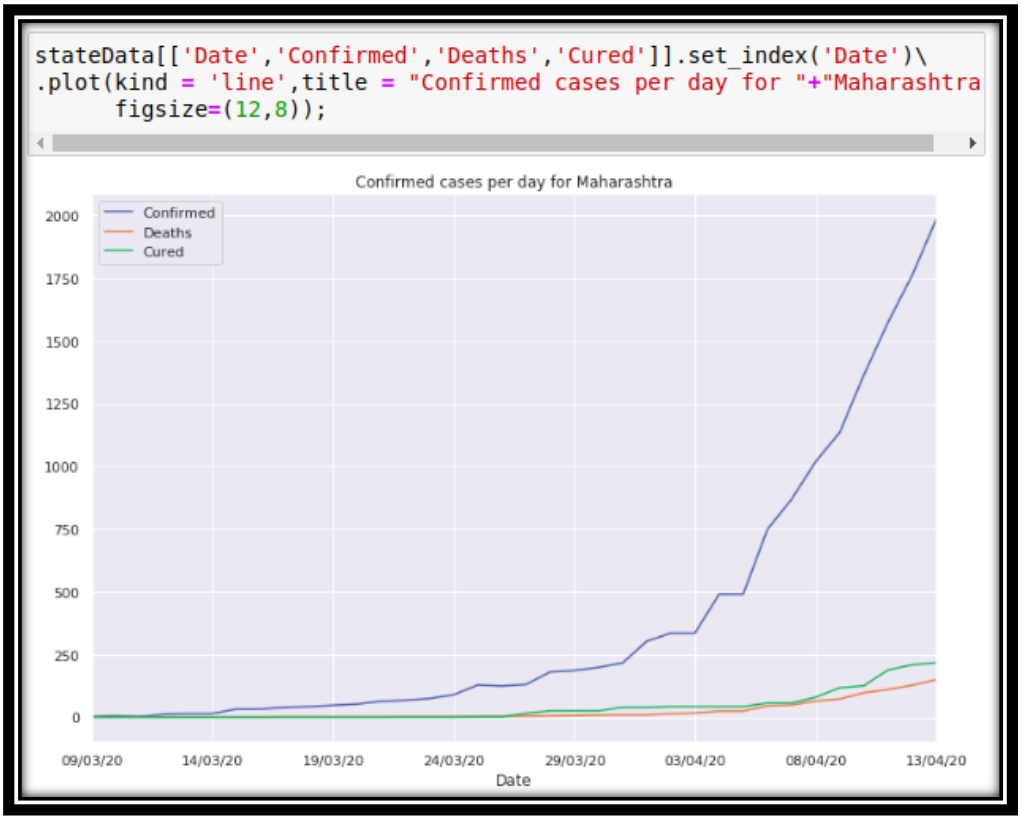


Figure 18

(ii) Analysing the per day increase in Confirmed Cases :

In this case, instead of plotting the per day confirmed cases, we will study the change in cases per day or in simpler words, the increase in the number of cases every day for Maharashtra. For this, we will add a new column to our data frame under consideration. This can be done by the following code snippet :

```
perDayIncrement = []
for ind,val in enumerate(stateData['Confirmed']):
    if ind == 0 or ind == len(stateData['Confirmed'] - 1):
        perDayIncrement.append(val)
    else:
        incre = (val - stateData['Confirmed'].iloc[ind-1])
        perDayIncrement.append(incre)
else:
    stateData['perDayIncrement'] = perDayIncrement
```

Figure 19

Next we will store the columns Date, Confirmed and perDayIncrement into another pandas dataframe named as dfPerDay. Next we will plot perDayIncrement against Date.

```
dfPerDay = stateData[['Date','Confirmed',"perDayIncrement"]]
dfPerDay[['Date','perDayIncrement']].set_index('Date')\
.plot(kind = 'line',title = "Per Day Increment for "+
      "Maharashtra",figsize=(12,8))
plt.show()
```

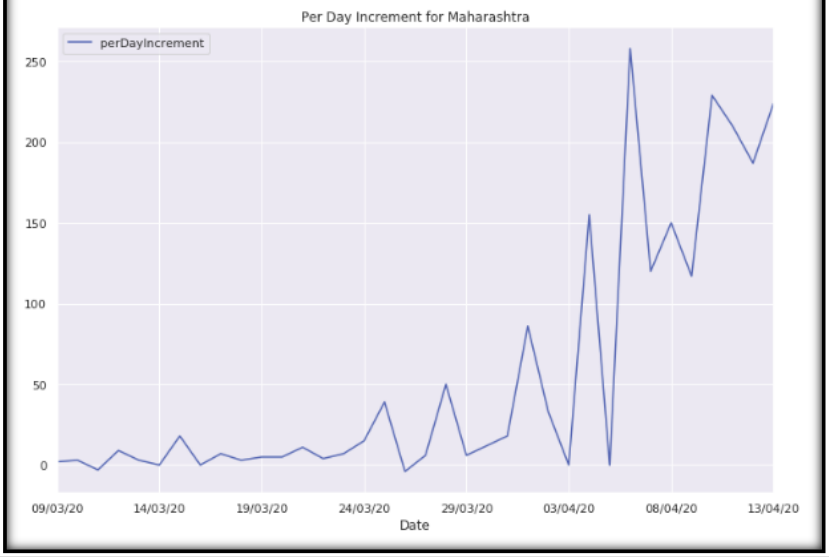


Figure 20

(iii) Studying the Box Plot

The concept of Box plot is explained in Chapter 3. Here we will study the Box plot for perDayIncrement cases of Maharashtra. It is implemented using the seaborn library as shown below. The diamonds represent the outliers encountered in the graph.

```
sns.boxplot(y = dfPerDay['perDayIncrement'] ,orient='v',
            palette="Blues").set_title('Box Plot for '+"Maharashtra")
Text(0.5, 1.0, 'Box Plot for Maharashtra')
```

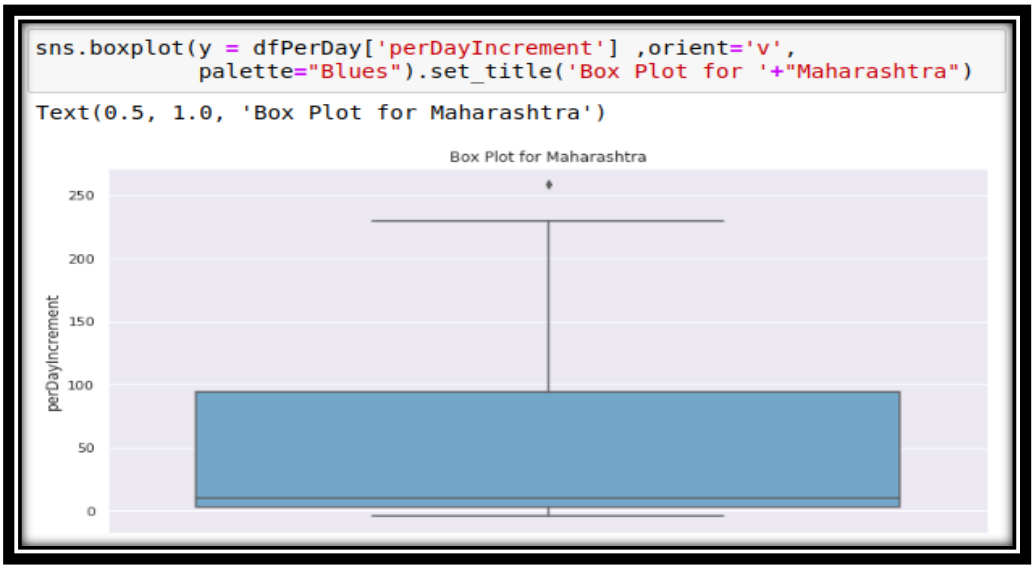


Figure 21

The State wise analysis is done for each state using this data set and all this analysis will be revisited in chapter 5 when we will build a prediction model for each state. There we will handle the skewness of the data and the outliers in the box plot. But as the scope of this chapter is concerned, we will move onto our next dataset for consideration.

3. HospitalBedsIndia.csv

This dataset consists about details of types of Hospitals and Hospital Beds in India in different states and their number count in each state and union territory. Let us see the structure of data.

```
hospitalData = pd.read_csv('HospitalBedsIndia.csv')
hospitalData = hospitalData.drop('Sno',axis = 1)
print(hospitalData.shape)
hospitalData.head()
```

```
(37, 11)
```

	State/UT	NumPrimaryHealthCenters_HMIS	NumCommunityHealthCenters_HMIS	NumSubDistric
0	Andaman & Nicobar Islands	27	4	
1	Andhra Pradesh	1417	198	
2	Arunachal Pradesh	122	62	
3	Assam	1007	166	
4	Bihar	2007	63	

Figure 22

The data has structure of 27 rows and 11 columns. The different types of hospitals covered over the Indian region as given below :

1. NumPrimaryHealthCenters_HMIS,

2. NumCommunityHealthCenters_HMIS,

3. NumSubDistrictHospitals_HMIS,

4. NumDistrictHospitals_HMIS,

5. TotalPublicHealthFacilities_HMIS,

6. NumPublicBeds_HMIS,

7. NumRuralHospitals_NHP18,

8. NumRuralBeds_NHP18,

9 NumUrbanHospitals_NHP18,

10. NumUrbanBeds_NHP18

Next, we check for any null cells in the data and get their index. Since this data is about the number of hospitals and involves infrastructure, so applying data cleaning techniques such as filling the values ourselves or using methods of measuring central tendency such as mean, median etc. Will not work here as we are taking about exact numbers of a physical Infrastructure. So, we will get the indexes of the null spaces and will enter the value -1 in them so that they won't appear in our pie chart analysis. This is done as :

```
hospitalData.isnull().any(1).nonzero()
```

```
(array([ 0,  2,  8, 14, 22, 24, 33]),)
```

```
hospitalData = hospitalData.fillna(-1)
hospitalData.head()
```

yHealthCenters_HMIS	NumSubDistrictHospitals_HMIS	NumDistrictHospitals_HMIS	TotalPublicHealthF
4	-1.0	3	
198	31.0	20	

Figure 23

Now with this dataset, we can get an idea about the percentage of different types of hospitals in the country and even this analysis can be taken for individual states. For this we will use Pie chart from the Plotly library. We will Implement this using a for loop for each state/UT and plot their respective Pie chart.

One such Example for Andaman & Nicobar Islands and Tamil Nadu are shown below :

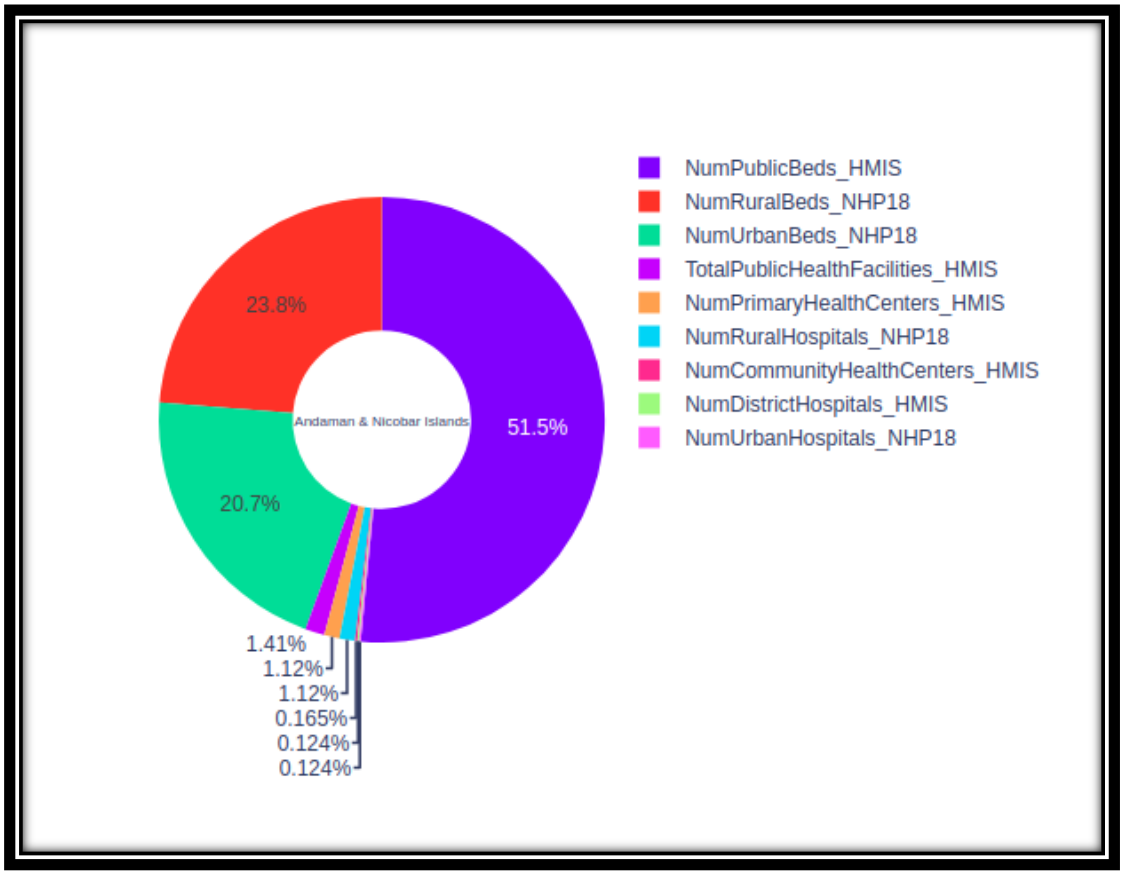


Figure 24

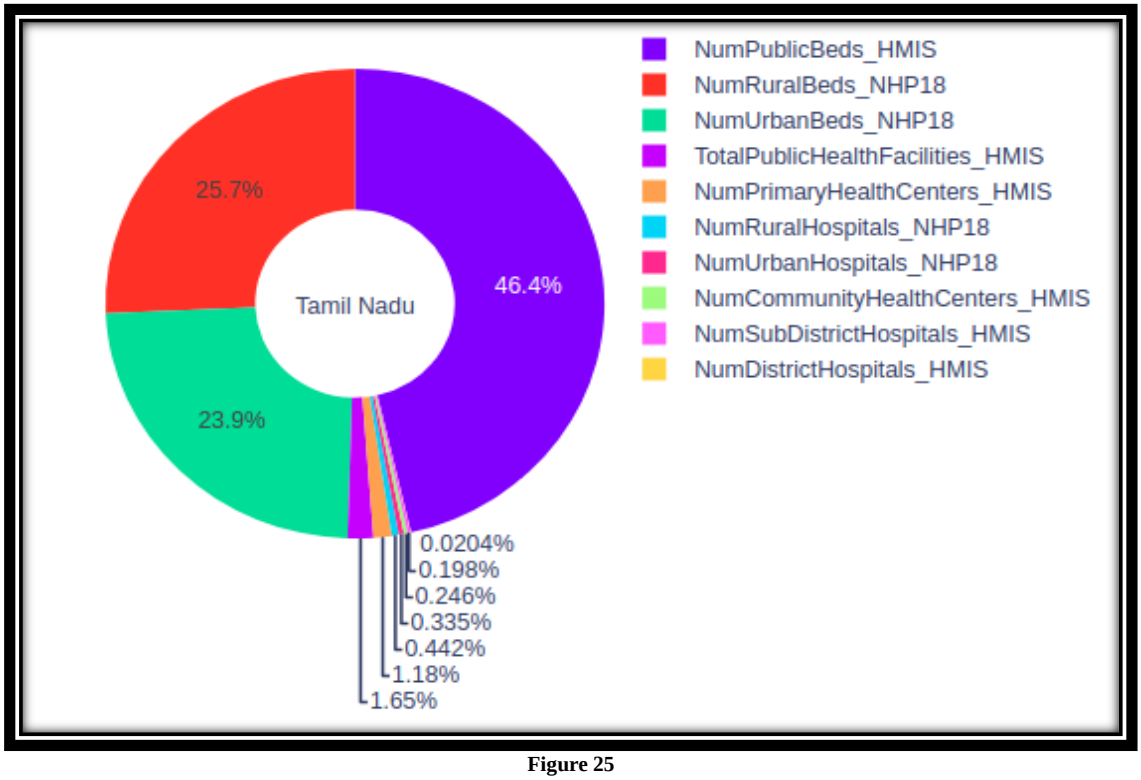


Figure 25

Pie charts for other states and UTs are given at the End of this Report. Currently only 2 Pie charts are shown for reference.

Next, we will study the count of each category mentioned about all over India. For this, we will be using a Bar plot from the Seaborn library

```
#Different Hospital Counts in India

cols = list(hospitalData.columns)
cols.remove('State/UT')
SumData = hospitalData[cols].iloc[:-1,:].sum(axis=0,skipna=True)
bplot = sns.barplot(x = SumData.index,y = SumData,palette="twilight" );
bplot.set_xticklabels(rotation=-90,labels= SumData.index);
bplot.set_title("Count of Different Categories of Hospitals in India !");
```

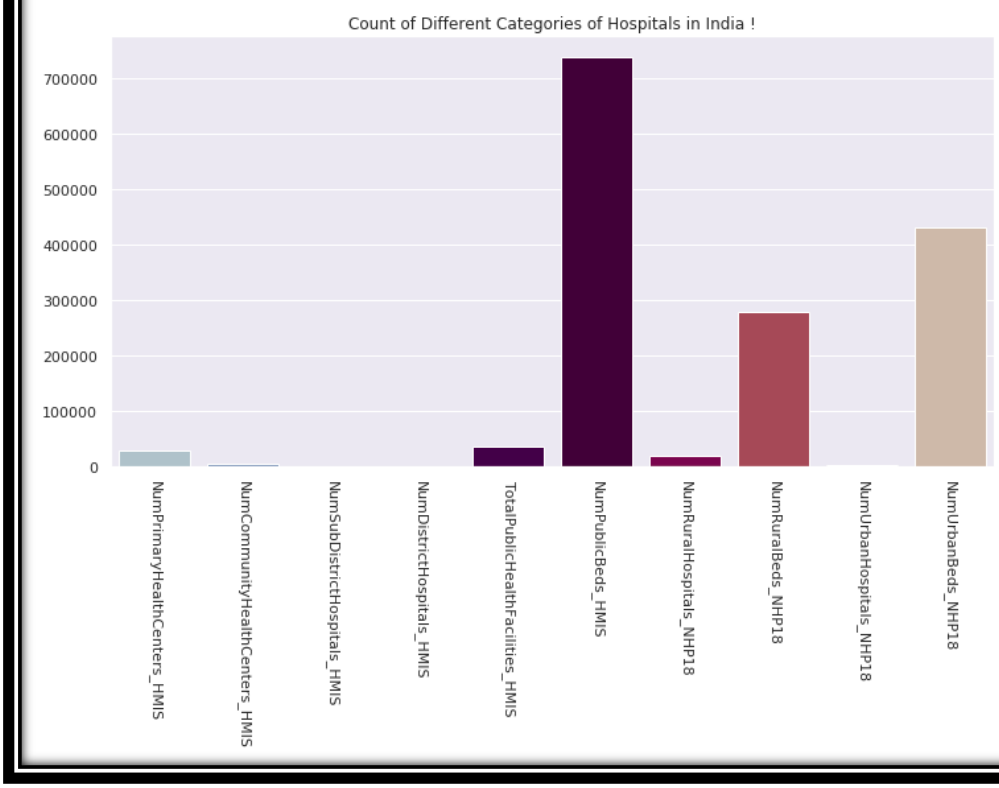


Figure 26

4. ICMRTestingLabs.csv

This dataset consists of details about the Corona Testing Labs spread over India in different states/UT. To understand the data lets load it using pandas Data frame and output the first 5 entries of the data frame.

```
labData = pd.read_csv('ICMRTestingLabs.csv')
labData.head()
```

	lab	address	pincode	city	state	type
0	ICMR-Regional Medical Research Centre, Port Blair	ICMR-Regional Medical Research Centre, Post Ba...	744103	Port Blair	Andaman and Nicobar Islands	Government Laboratory Supported
1	Tomo Riba Institute of Health & Medical Scienc...	National Highway 52A, Old Assembly Complex, Na...	791110	Naharlagun	Arunachal Pradesh	Collection Site
2	Sri Venkateswara Institute of Medical Sciences...	Sri Venkateswara Institute of Medical Sciences...	517507	Tirupati	Andhra Pradesh	Government Laboratory Supported
3	Rangaraya Medical College, Kakinada	Rangaraya Medical College, Kakinada Pithampura...	533001	Kakinada	Andhra Pradesh	Government Laboratory Supported
4	Sidhartha Medical College, Vijaywada	Siddhartha Medical College, Vijayawada NH 16 S...	520008	Vijayawada	Andhra Pradesh	Government Laboratory Supported

Figure 27

The dataset consists of 5 columns, namely, lab (it's name), address, pincode, city, state and type.

For the Analysis part, we can extract the number of labs in each state, and another thing that can be analysed from the data can be count of different types of labs over India. Lets see the different categories of Labs :

1. Government Laboratory Supported,

2. Collection Site,

3. Private Laboratory,

4. Government Laboratory Suitable

Next let us see the count of labs in each state. For this we will group the data by state and store the count of each lab.
This is shown as below :

```
stateLabs = labData.groupby('state')[['lab']].aggregate('count')

stateLabs.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 34 entries, Andaman and Nicobar Islands to West Bengal
Data columns (total 1 columns):
lab    34 non-null int64
dtypes: int64(1)
memory usage: 544.0+ bytes
```

Figure 28

Let us analyse this with the help of the Bar Plot. Here we plot the count of labs against their state to know how many labs are in each state.

```
labChart = sns.barplot(x = stateLabs.index,y = stateLabs.lab)
labChart.set_xticklabels(rotation = -90, labels = stateLabs.index)
labChart.set_title("Count of Number of Testing Labs Across India (State Wise)");
plt.show()
```

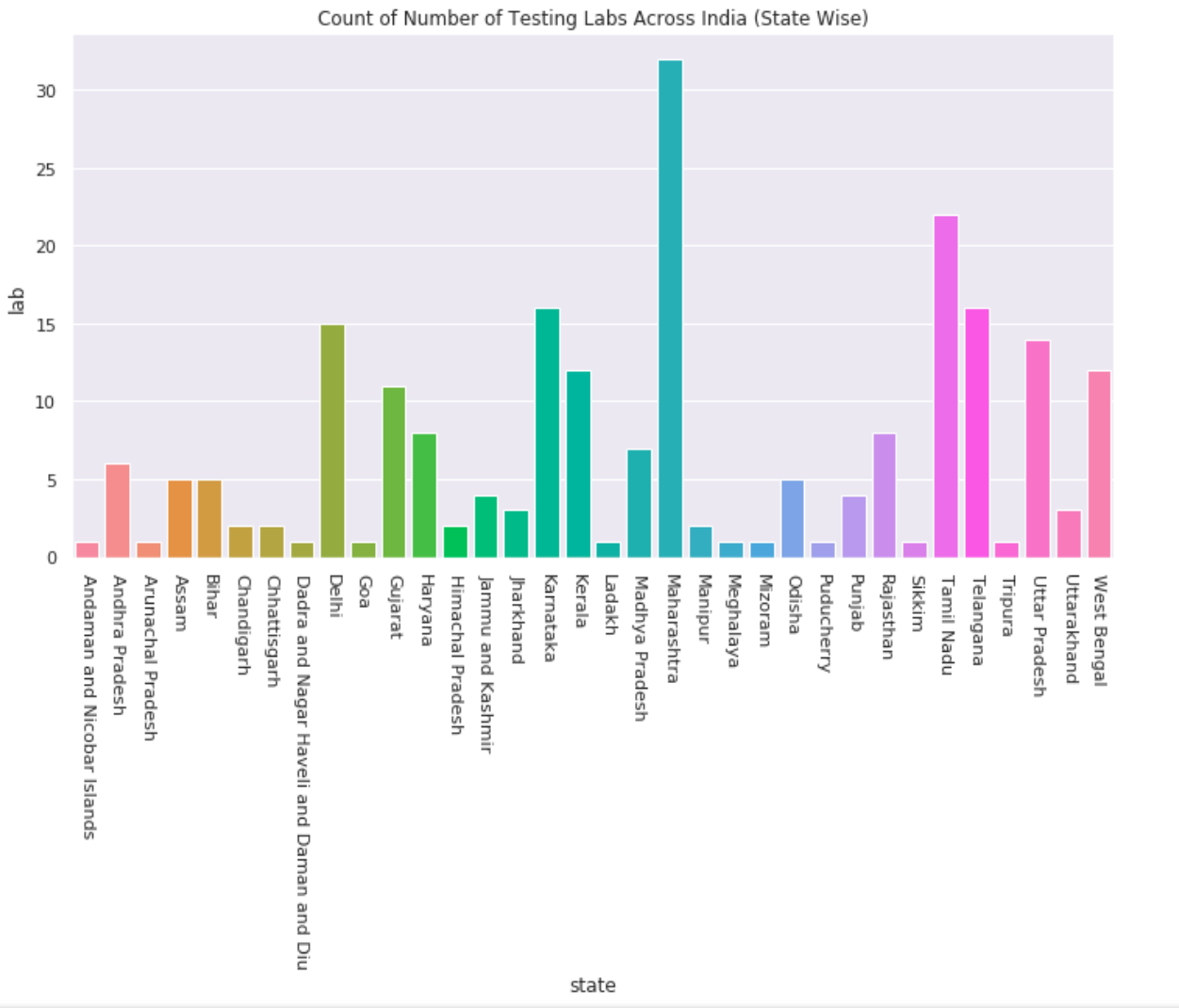


Figure 29

Next we will analyse the Count of different types of labs as mentioned above, across the Indian region with the help of a Bar plot. For that, we will first make a separate Data Frame by grouping the data based on type and counting the number of labs for different types.

```
stateLabCategory = labData.groupby('type')[['lab']].aggregate('count')
stateLabCategory.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 4 entries, Collection Site to Private Laboratory
Data columns (total 1 columns):
lab    4 non-null int64
dtypes: int64(1)
memory usage: 64.0+ bytes
```

Figure 30

Then this data frame is used to plot the count against different types of labs as shown below :

```
labCategory = sns.barplot(x = stateLabCategory.index,y = stateLabCategory.lab,palette='inferno_r')
labCategory.set_xticklabels(rotation=30,labels = stateLabCategory.index)
labCategory.set_title("Count of Type of Testing Labs in India");
plt.show()
```

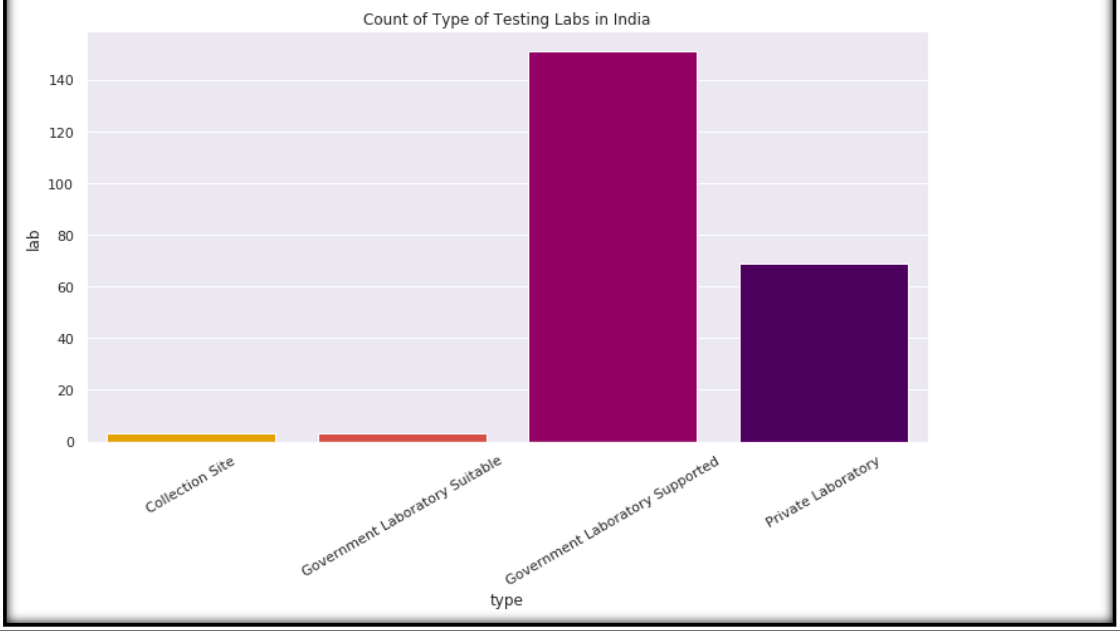


Figure 31

5. ICMRTestingDetails.csv

This dataset consists of the Testing details being carried out in our country as posted by Indian Council of Medical Research (ICMR). Let us dive into the understanding of the dataset by describing its Numerical Attributes :

```
testingDetails = pd.read_csv('ICMRTestingDetails.csv')
```

```
testingDetails.describe()
```

	SNo	TotalSamplesTested	TotalIndividualsTested	TotalPositiveCases
count	27.000000	24.000000	13.000000	22.000000
mean	14.000000	67858.333333	59130.307692	2686.636364
std	7.937254	58280.552277	68175.957192	2689.697567
min	1.000000	6500.000000	5900.000000	78.000000
25%	7.500000	20062.000000	14514.000000	414.750000
50%	14.000000	45369.500000	19817.000000	1846.500000
75%	20.500000	104304.750000	130792.000000	4495.750000
max	27.000000	195748.000000	181028.000000	8312.000000

Figure 32

Next, we will look into some values of the dataset and found out that this contains Missing values. Now since these are testing details, using Measures of Central Tendency won't be a good option. So, we will get rid of the missing values and the Sno column. The resulting set of values are shown below :

```
testingDetails.tail()
```

	SNo	DateTime	TotalSamplesTested	TotalIndividualsTested	TotalPositiveCases	Source
22	23	08/04/20 21:00	127919.0	NaN	5114.0	NaN
23	24	09/04/20 21:00	144910.0	130792.0	5705.0	NaN
24	25	10/04/20 21:00	161330.0	147034.0	6872.0	NaN
25	26	11/04/20 21:00	179374.0	164773.0	7703.0	NaN
26	27	12/04/20 21:00	195748.0	181028.0	8312.0	NaN

```
testingDetails = testingDetails.drop('Source',axis=1)
```

```
testingDetails.dropna(axis=0,inplace=True)
testingDetails
```

	SNo	DateTime	TotalSamplesTested	TotalIndividualsTested	TotalPositiveCases
0	1	13/03/20 18:00	6500.0	5900.0	78.0
1	2	18/03/20 18:00	13125.0	12235.0	150.0
2	3	19/03/20 18:00	14175.0	13285.0	182.0
3	4	20/03/20 18:00	15404.0	14514.0	236.0
4	5	21/03/20 18:00	16911.0	16021.0	315.0
5	6	22/03/20 18:00	18127.0	17237.0	396.0
6	7	23/03/20 20:00	20707.0	19817.0	471.0
7	8	24/03/20 20:00	22694.0	21804.0	536.0
8	9	25/03/20 20:00	25144.0	24254.0	581.0
23	24	09/04/20 21:00	144910.0	130792.0	5705.0
24	25	10/04/20 21:00	161330.0	147034.0	6872.0
25	26	11/04/20 21:00	179374.0	164773.0	7703.0
26	27	12/04/20 21:00	195748.0	181028.0	8312.0

Figure 33

Next, we will use this Data Frame to study the Increase in Positive cases with the increase in Individual Testing. This is shown below :

```
inc = sns.barplot(x =testingDetails['TotalIndividualsTested'],y=testingDetails['TotalPositiveCases'],palette="inferno");
inc.set_title("Bar Plot Showing Variation of Positive Cases with increased Testing in India");
inc.set_xticklabels(labels = testingDetails['TotalIndividualsTested'],rotation = -30);
```

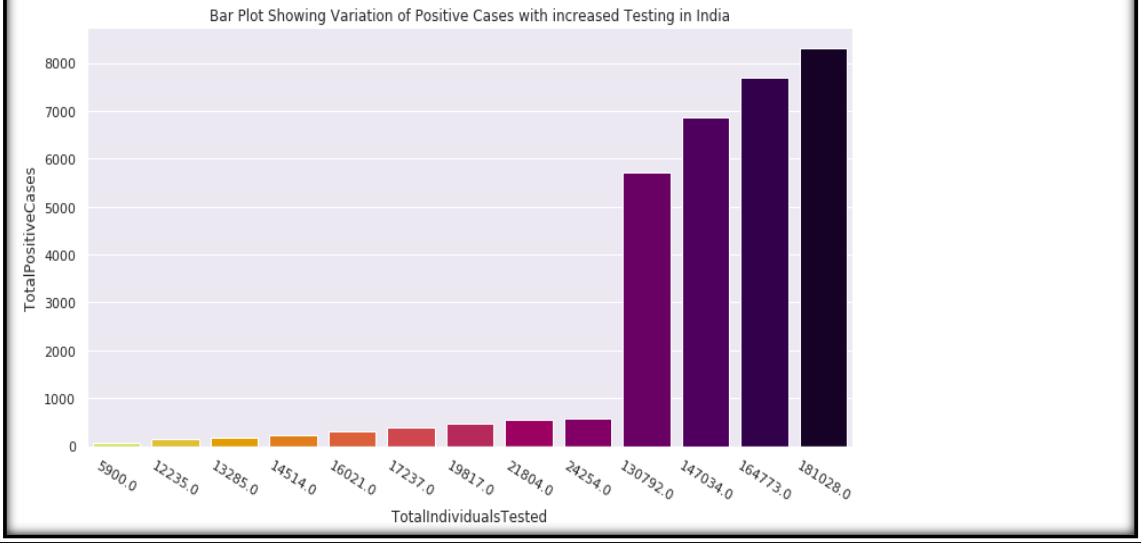


Figure 34

6. StatewiseTestingDetails.csv

As the name suggest, this data set consists details about state level testing and how many total tests were conducted, the positive cases and the negative cases. Let us describe the numeric content of this data set.

```
stateTests = pd.read_csv('StatewiseTestingDetails.csv')
stateTests.columns
```

```
Index(['Date', 'State', 'TotalSamples', 'Negative', 'Positive'], dtype='object')
```

```
stateTests.describe()
```

	TotalSamples	Negative	Positive
count	201.000000	198.000000	201.000000
mean	3960.955224	3518.702020	171.805970
std	5183.873400	4873.665925	245.123861
min	49.000000	49.000000	0.000000
25%	731.000000	608.250000	9.000000
50%	2297.000000	1851.000000	72.000000
75%	5015.000000	4071.500000	234.000000
max	31841.000000	30477.000000	1364.000000

Figure 35

Next, we check for the Null values (if any). The Negative column has 3 null values and since it's the testing data so applying a measure of central tendency won't be a good choice, so we remove the null values from the Dataset.

```
stateTests.dropna(axis=0,inplace=True)
stateTests.isnull().sum()
```

```
Date            0
State           0
TotalSamples    0
Negative        0
Positive        0
dtype: int64
```

Figure 36

Next we will plot the horizontal Bar Graph for the state wise analysis for total tests done, positives and negatives for each state in the data set.

```
stateTests.groupby('State')[['TotalSamples','Negative','Positive']].aggregate(max)\
.plot(kind='barh',stacked=True,title="State Wise Testing Statistics (Stacked)",figsize=(12,12))
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f6989674ed0>
```

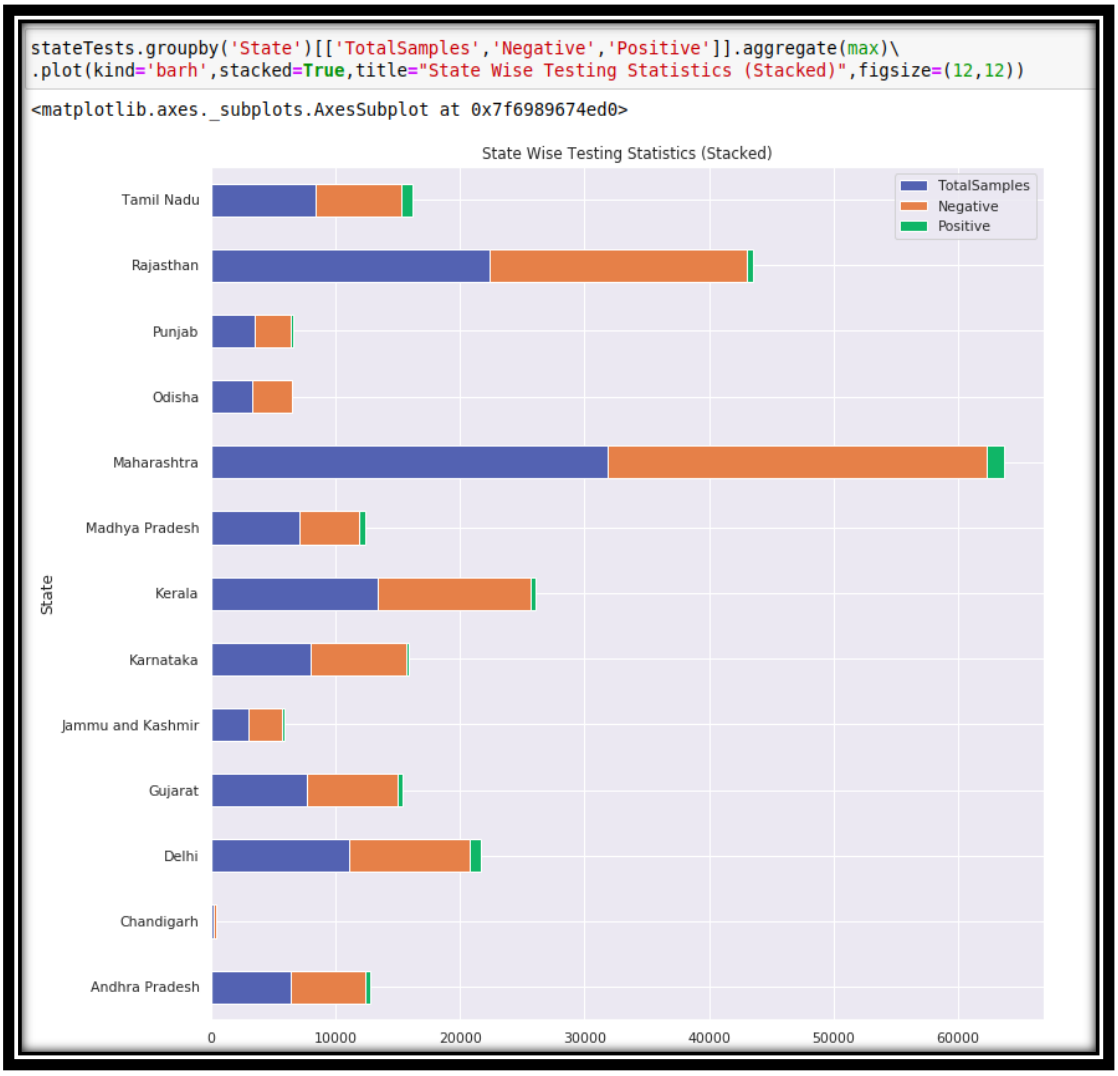


Figure 37

```
stateTests.groupby('State')[['TotalSamples','Negative','Positive']].aggregate(max)\
.plot(kind='barh',title="State Wise Testing Statistics",figsize=(10,15))
```

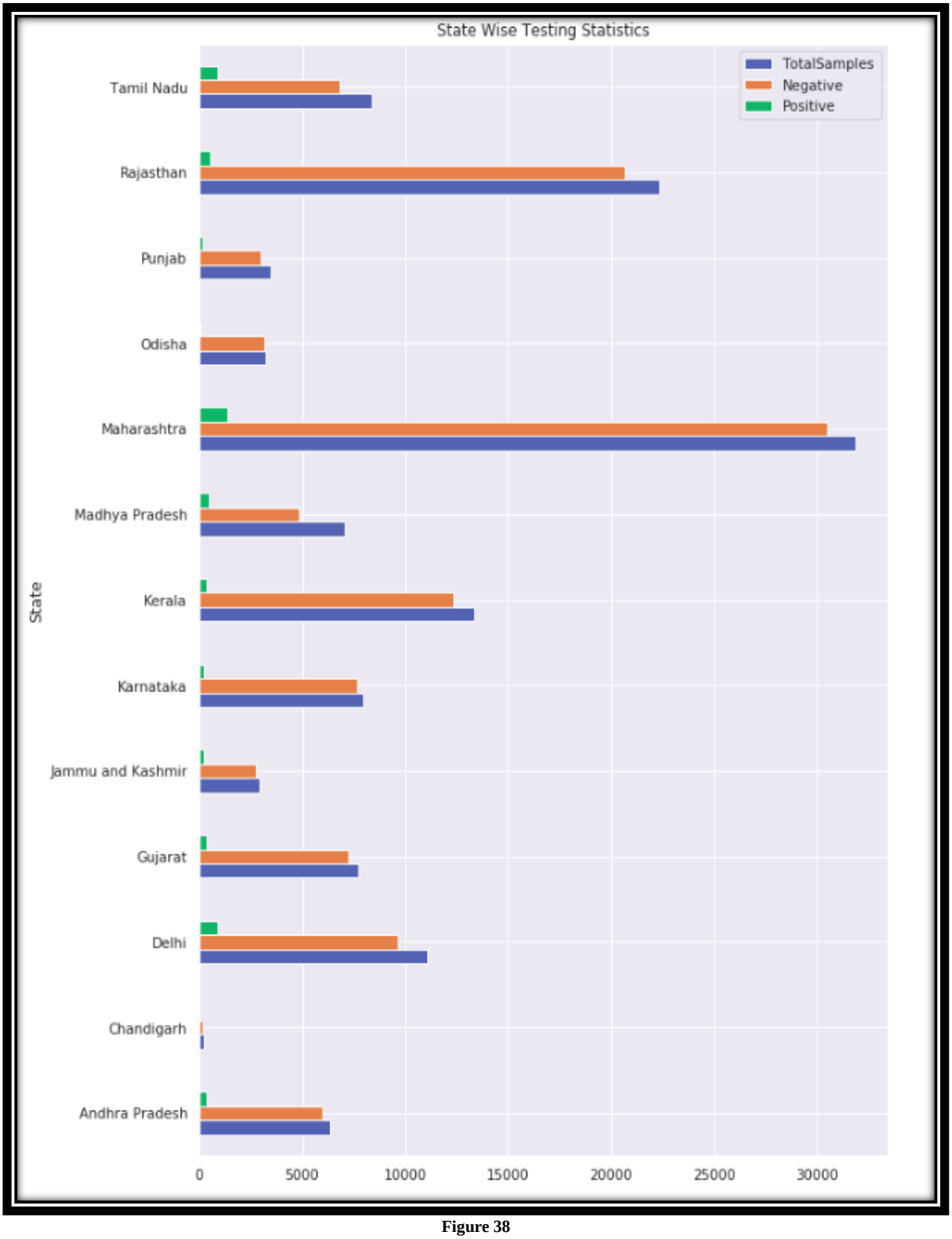


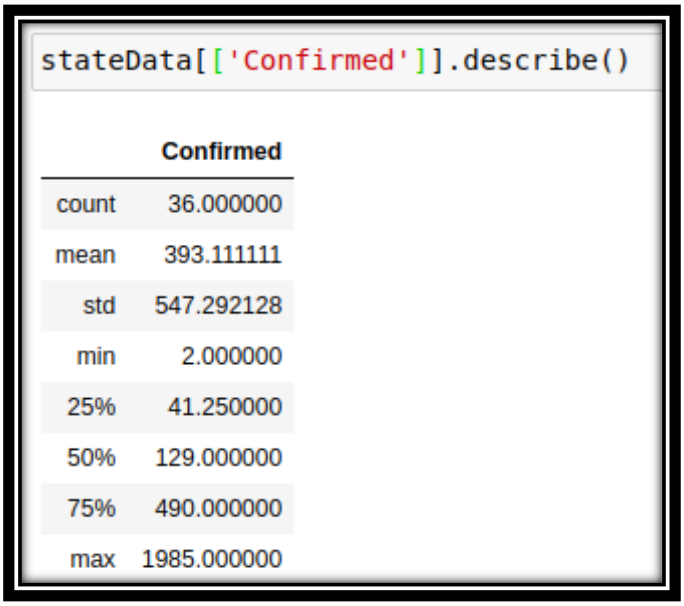
Figure 38

CHAPTER 5 : PREDICTION ANALYSIS

In Chapter 4, we studied the analysis of all the data sets being used in this assignment , their attributes, and extracted useful information to as much extent as possible, and visualized them using variety of plots. Now in this chapter, we will try to do prediction analysis. Recall our work on the state of Maharashtra in chapter 4, we will be making a model that will predict the Number of Confirmed Cases on current day's date for a particular state, based on the data from the previous days.

To begin the understanding of the prediction model , lets recall our work for the state of Maharashtra. Here we will be explaining the process for one state only and in the end of this chapter, all the plots discussed for all the states will be pasted. So now recalling to our previous work , we did till understanding the Boxplot for Maharashtra.

Now let us understand the confirmed data by describing some useful attributes.



```
stateData['Confirmed'].describe()
```

	Confirmed
count	36.000000
mean	393.111111
std	547.292128
min	2.000000
25%	41.250000
50%	129.000000
75%	490.000000
max	1985.000000

Figure 39

It is visible that the data is highly skewed, as the minimum value is 2 and that grows upto 1985 in just 36 entries of data. There is a large difference between the Mean (at 393.11) and the Median (at 129) which double confirms the skewness of the data.

So the prediction analysis can be handled in the following way.

Working on per Day Increment

Now If we try to understand what is going on, if it was possible to decrease the range of growth of the confirmed cases and then applying some skewness measures, we might get a much better and optimal fit for the data. But the problem arises, how to do this ? One such possible way is that instead of taking Confirmed Cases for Each day, we will take the per day increment in our Consideration. The per day increment was analysed in Chapter 4 and here we will try to fit our Linear Model into the Less skewed data.

```
fig = go.Figure()
dfPerDay['Date'] = pd.to_datetime(dfPerDay['Date'],dayfirst=True)
dfPerDay['Date'] = dfPerDay['Date'].map(dt.datetime.toordinal)

x = dfPerDay['Date'].values
x = x.reshape(-1,1)
y = dfPerDay['perDayIncrement'].values

rfmodel = LinearRegression(fit_intercept=True)

rfmodel.fit(x,y)
y_predition = rfmodel.predict(x)

fig.add_trace(go.Scatter(x = dfPerDay['Date'] , y= dfPerDay['perDayIncrement'],
                    mode='markers', name='Actual Points'))

fig.add_trace(go.Scatter(x=dfPerDay['Date'], y=y_predition,
                    mode='lines+markers',
                    name='Regression Fit'))
mse = mean_squared_error(y,y_predition)
print("Mean Squared Error : "+str(mse))
fig.show()
```

```
Mean Squared Error : 2221.0916107552057
```

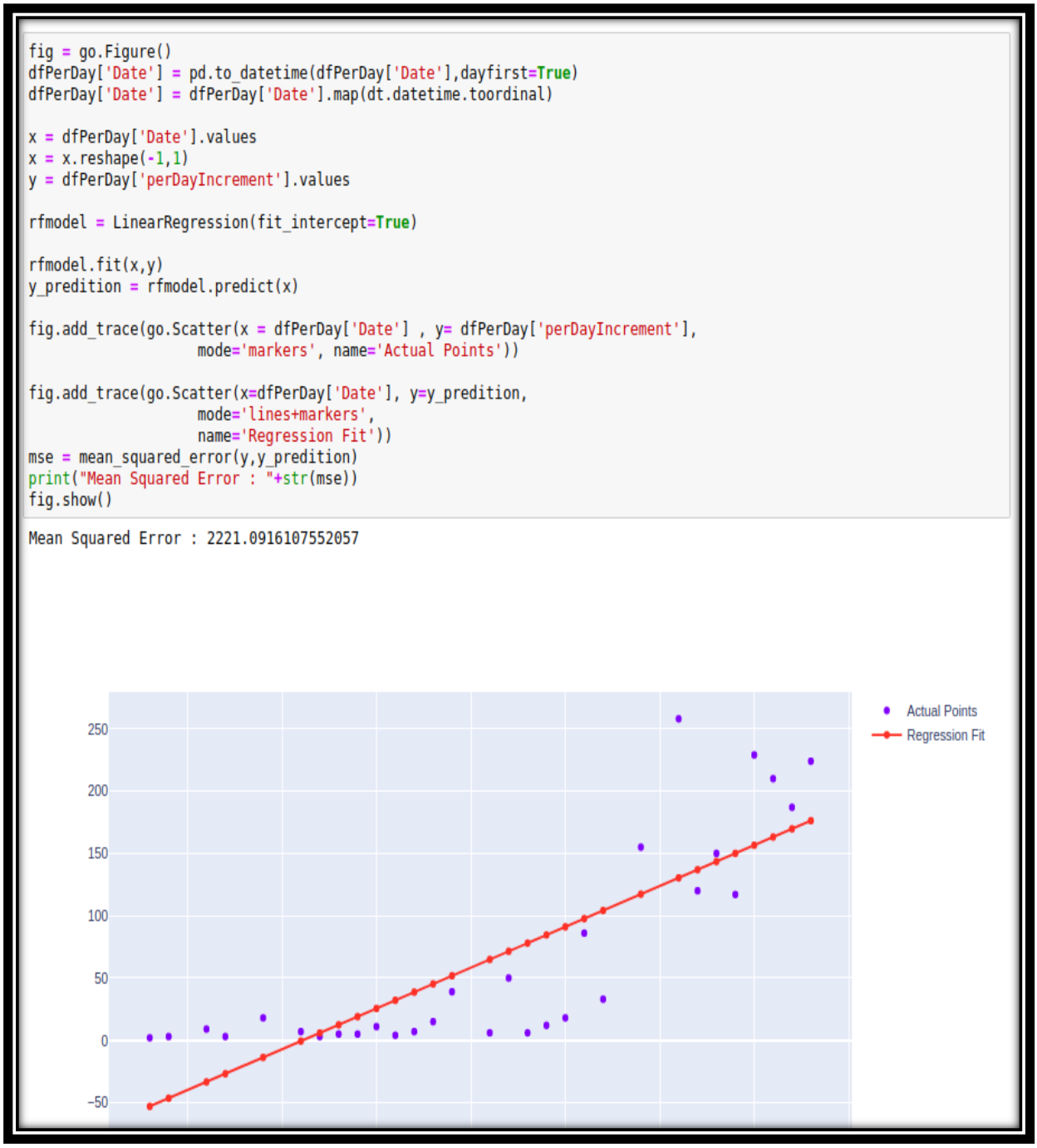


Figure 40

This is the fitting of our Line, without any Skewness measure. We can see that we have a high Mean Squared Error Function. Next we will take Cube root and then we will take Log and understand the things better.

Skewness measure :

1. Cube Root Function

2.Log Function

Cube Root Function

With a Mean Square Error of 1.73 , it is a better fit as compared to the Above Case. This per day increase can be added to the previous date value to get the prediction for present day max confirmed Cases.

```
fig = go.Figure()
dfPerDay['Date'] = pd.to_datetime(dfPerDay['Date'],dayfirst=True)
dfPerDay['Date'] = dfPerDay['Date'].map(dt.datetime.toordinal)
dfPerDay['perDayIncrement'] = dfPerDay['perDayIncrement'].apply(lambda x: (x**(1/3)).real )

x = dfPerDay['Date'].values
x = x.reshape(-1,1)
y = dfPerDay['perDayIncrement'].values

rfmodel = LinearRegression(fit_intercept=True)

rfmodel.fit(x,y)
y_predition = rfmodel.predict(x)

fig.add_trace(go.Scatter(x = dfPerDay['Date'] , y= dfPerDay['perDayIncrement'],
                    mode='markers', name='Actual Points'))

fig.add_trace(go.Scatter(x=dfPerDay['Date'], y=y_predition,
                    mode='lines+markers',
                    name='Regression Fit'))
mse = mean_squared_error(y,y_predition)
print("Mean Squared Error : "+str(mse))
fig.show()
```

```
Mean Squared Error : 1.7341082184799097
```

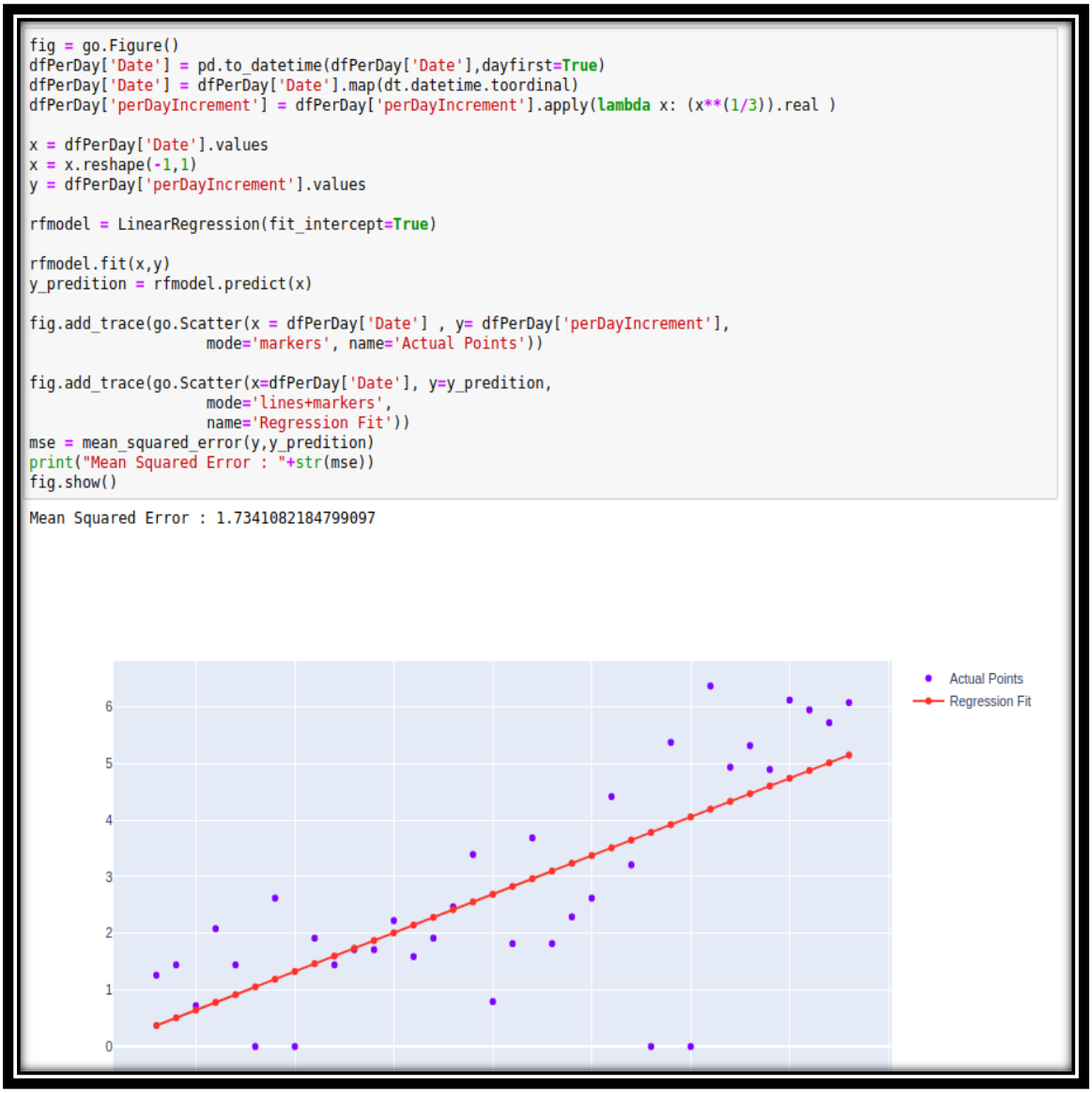


Figure 41

Log Function

Next, we will be using Log Function to decrease the skewness of the data points.

It was Observed that Calculating the Per Day increase and adding it to the Previous Date's Value Gave a better Prediction as in the Case of Calculating the Confirmed case value for a Day.

```
fig = go.Figure()
dfPerDay['Date'] = pd.to_datetime(dfPerDay['Date'],dayfirst=True)
dfPerDay['Date'] = dfPerDay['Date'].map(dt.datetime.toordinal)
dfPerDay['perDayIncrement'] = np.log(dfPerDay['perDayIncrement'])

x = dfPerDay['Date'].values
x = x.reshape(-1,1)
y = dfPerDay['perDayIncrement'].values

rfmodel = LinearRegression(fit_intercept=True)

rfmodel.fit(x,y)
y_predition = rfmodel.predict(x)

fig.add_trace(go.Scatter(x = dfPerDay['Date'] , y= dfPerDay['perDayIncrement'],
                    mode='markers', name='Actual Points'))

fig.add_trace(go.Scatter(x=dfPerDay['Date'], y=y_predition,
                    mode='lines+markers',
                    name='Regression Fit'))
mse = mean_squared_error(y,y_predition)
print("Mean Squared Error : "+str(mse))
fig.show()
```

```
Mean Squared Error : 0.5288603436496292
```

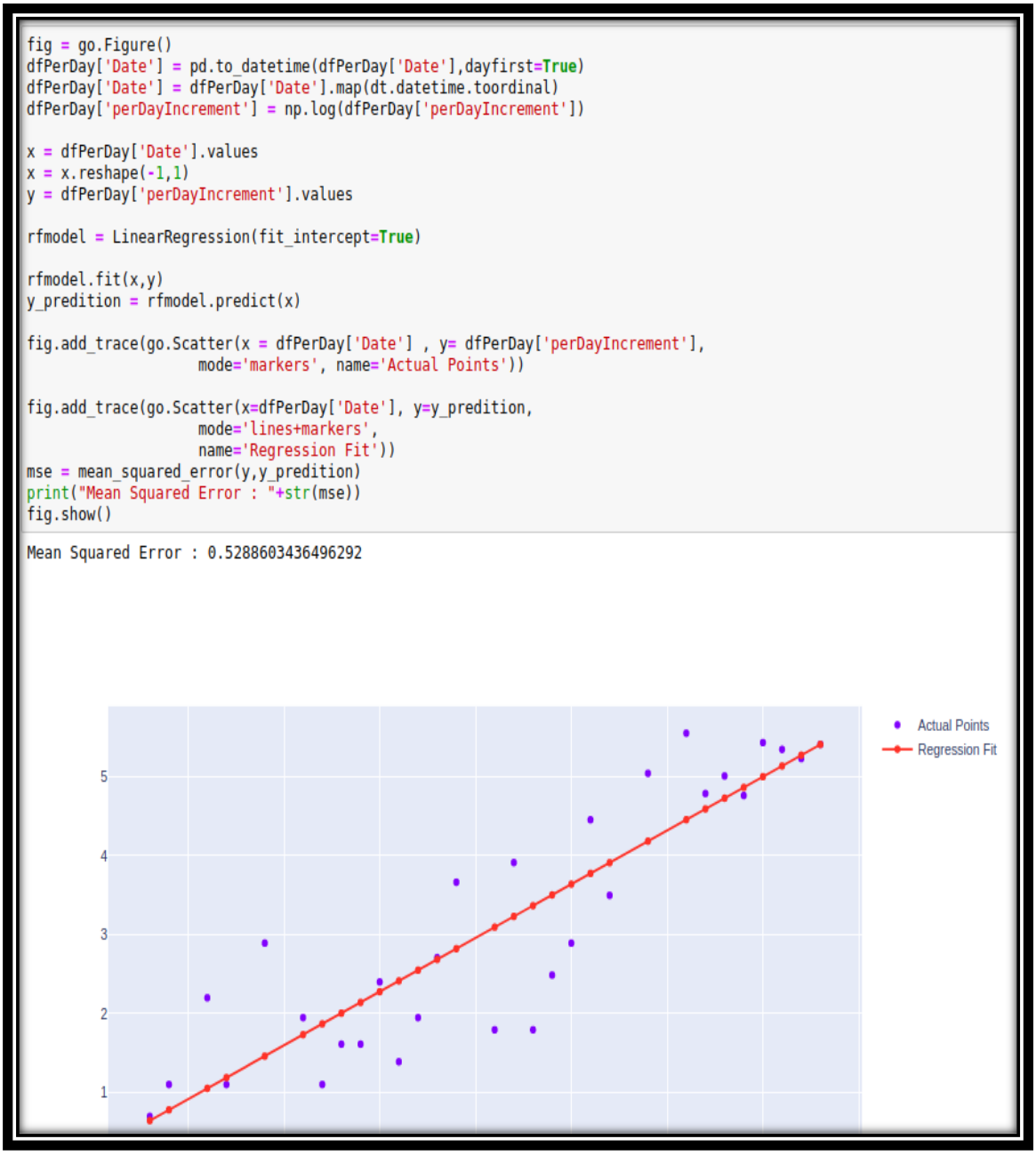


Figure 42

Log function gives the best and optimal fitting of the data whose prediction values are nearly meeting the Actual Values.

Case of Overfitting

This is a special case, when the data for prediction was per day confirmed cases instead of per day increment. So the data was studied and was given to the Linear model, and by using the Log Function to control the Skewness of the data , it was observed that it gave the best fit with least Mean Square Error, by passing almost every value of the Data Points. However this gave rise to overfitting, and it was observed that it worked correctly for the training data set, however it work poorly for the Test Data. This Case is shown below :

```
fig = go.Figure()
dfPerDay['Date'] = pd.to_datetime(dfPerDay['Date'],dayfirst=True)
dfPerDay['Date'] = dfPerDay['Date'].map(dt.datetime.toordinal)
dfPerDay['Confirmed'] = np.log(dfPerDay['Confirmed'])

x = dfPerDay['Date'].values
x = x.reshape(-1,1)
y = dfPerDay['Confirmed'].values

rfmodel = LinearRegression(fit_intercept=True)

rfmodel.fit(x,y)
y_predition = rfmodel.predict(x)

fig.add_trace(go.Scatter(x = dfPerDay['Date'] , y= dfPerDay['Confirmed'],
                    mode='markers', name='Actual Points'))

fig.add_trace(go.Scatter(x=dfPerDay['Date'], y=y_predition,
                    mode='lines+markers',
                    name='Regression Fit'))
mse = mean_squared_error(y,y_predition)
print("Mean Squared Error : "+str(mse))
fig.show()
```

```
Mean Squared Error : 0.08975001900308786
```

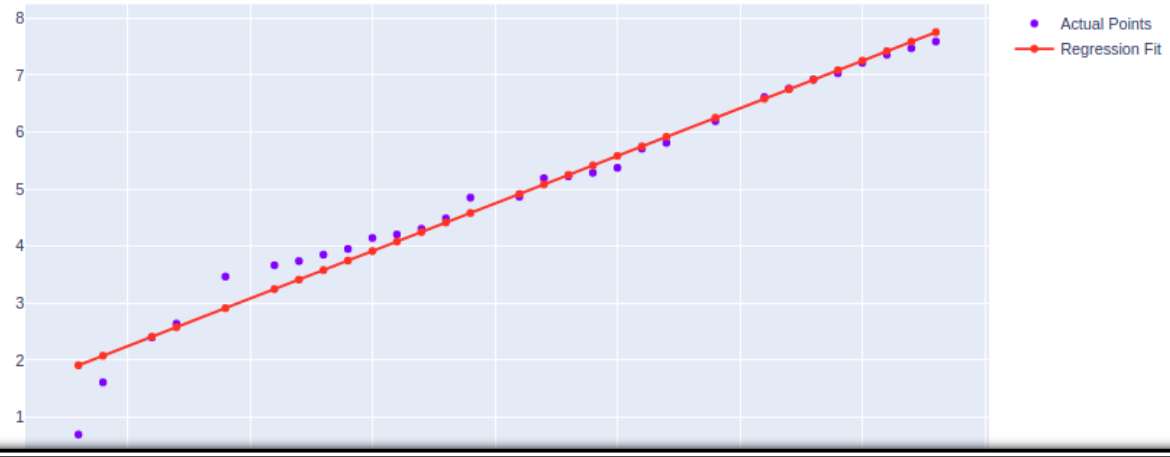


Figure 43

It was observed to be the best fitting over the training set , however its performance was poor on the Test Data , which is a property of overfit Data. So in order to deal with this, we worked upon the per Day increment of the Data.

Outlier Analysis :

Here we will see how Outliers were taken care of, when we wanted to train the model. This is because each data entry in this dataset is of significant importance and so it is our task to make sure that outliers donot effect our model training. This is achieved with the help of Box plots, we will first see the Boxplots for the confirmed cases everyday for Maharashtra and then for the per day increment in case.

Three Boxplots will be shown for each part,

1. Actual Boxplot
2. Cube root Applied Boxplot
3. Log Function applied Boxplot

Figure 44

In the first figure above, outliers can be observed as small diamonds shown.

```
f, axes = plt.subplots(1, 3)

sns.boxplot(y = dfPerDay['perDayIncrement'] ,orient='v' , ax=axes[0],
                    palette="Blues").set_title('Box Plot for Maharashtra');

sns.boxplot(y = dfPerDay['perDayIncrement'].apply(lambda x: (x**(1/3)).real ) , orient='v' , ax=axes[1],
            palette="Oranges").set_title('Box Plot '+' with Cube Root');

sns.boxplot(y = np.log(dfPerDay['perDayIncrement']) , orient='v' , ax=axes[2],
            palette="Dark2").set_title('Box Plot  with Log Function');
```

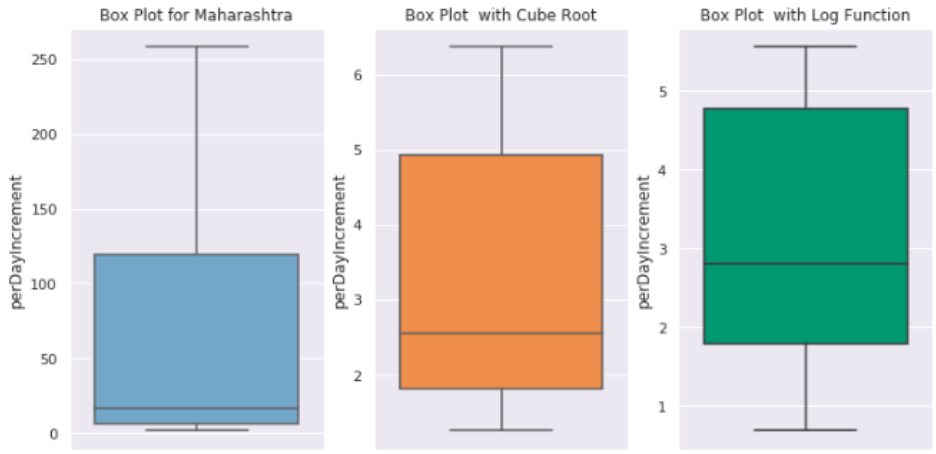


Figure 45

PREDICITON FOR ALL STATES :

This Model the we Developed using the Log Function for perDayIncrement is used to calculate the Confirmed Cases for each state that may reach in the present day. It is achieved using a for loop and then performing the Combined steps in Chapter 4 and Chapter 5

Please Turn Over

OUR PREDICTION RESULTS FOR 14th April 2020

State	Date	Confirmed Cases
Kerala	2020-04-14 00:00:00	392
Telengana	2020-04-14 00:00:00	627
Delhi	2020-04-14 00:00:00	1255
Rajasthan	2020-04-14 00:00:00	886
Uttar Pradesh	2020-04-14 00:00:00	515
Haryana	2020-04-14 00:00:00	199
Ladakh	2020-04-14 00:00:00	17
Tamil Nadu	2020-04-14 00:00:00	1228
Karnataka	2020-04-14 00:00:00	267
Maharashtra	2020-04-14 00:00:00	2249
Punjab	2020-04-14 00:00:00	185
Jammu and Kashmir	2020-04-14 00:00:00	276
Andhra Pradesh	2020-04-14 00:00:00	499
Uttarakhand	2020-04-14 00:00:00	42
Odisha	2020-04-14 00:00:00	61
Puducherry	2020-04-14 00:00:00	10
West Bengal	2020-04-14 00:00:00	170
Chhattisgarh	2020-04-14 00:00:00	36
Chandigarh	2020-04-14 00:00:00	24
Gujarat	2020-04-14 00:00:00	595
Himachal Pradesh	2020-04-14 00:00:00	41
Madhya Pradesh	2020-04-14 00:00:00	742
Bihar	2020-04-14 00:00:00	70
Manipur	2020-04-14 00:00:00	3
Andaman and Nicobar Islands	2020-04-14 00:00:00	12
Goa	2020-04-14 00:00:00	8
Assam	2020-04-14 00:00:00	32
Jharkhand	2020-04-14 00:00:00	24
Tripura	2020-04-14 00:00:00	3

Figure 46

ACTUAL RESULTS ON 14th April, 2020

** Courtesy : covid19india.org **

STATE/UT	CONFIRMED
MAHARASHTRA	2,334
DELHI	1,510
TAMIL NADU	1,173
RAJASTHAN	897
MADHYA PRADESH	614
TELANGANA	592
GUJARAT	572
UTTAR PRADESH	558
ANDHRA PRADESH	439
KERALA	378
JAMMU AND KASHMIR	270
KARNATAKA	247
HARYANA	196
PUNJAB	176
WEST BENGAL	152
BIHAR	66
ODISHA	55
UTTARAKHAND	35
HIMACHAL PRADESH	32
CHHATTISGARH	31
ASSAM	30
JHARKHAND	24
CHANDIGARH	21
LADAKH	17
ANDAMAN AND NICOBAR ISLANDS	11
GOA	7
PUDUCHERRY	7
MANIPUR	2
TRIPURA	2
ARUNACHAL PRADESH	1
DADRA AND NAGAR HAVELI	1
MEGHALAYA	1
MIZORAM	1
NAGALAND	1

Figure 47

ACTUAL RESULTS ON 14th April, 2020

** Courtesy : Ministry of Health and Family Welfare **

Andhra Pradesh	432
Andaman and Nicobar Islands	11
Arunachal Pradesh	1
Assam	31
Bihar	64
Chandigarh	21
Chhattisgarh	31
Delhi	1154
Goa	7
Gujarat	539
Haryana	185
Himachal Pradesh	32
Jammu and Kashmir	245
Jharkhand	19
Karnataka	247
Kerala	376
Ladakh	15
Madhya Pradesh	604
Maharashtra	1985
Manipur	2
Mizoram	1
Nagaland	1
Odisha	54
Puducherry	7
Punjab	167
Rajasthan	812
Tamil Nadu	1075
Telengana	562
Tripura	2
Uttarakhand	35
Uttar Pradesh	483
West Bengal	152

Figure 48

PREDICITION PLOTS FOR EACH STATE


Confirmed cases per day for Telengana
Confirmed
Deaths
Cured
500
400
300
200
100
0
02/03/20
07/03/20
12/03/20
17/03/20
22/03/20
27/03/20
01/04/20
06/04/20
11/04/20
Date
Per Day Increment (w/o Log) for Telengana
perDayIncrement
100
80
60
40
20
0
02/03/20
21/03/20
27/03/20
01/04/20
06/04/20
11/04/20
Date
Box Plot for Telengana without Log
perDayIncrement
100
80
60
40
20
0
Box Plot for Telengana with Log
perDayIncrement
4
3
2
1
0
Per Day Increment (Log applied) for Telengana
perDayIncrement
4
3
2
1
0
02/03/20
21/03/20
27/03/20
01/04/20
06/04/20
11/04/20
Date


Confirmed cases per day for Delhi
Confirmed
Deaths
Cured
1200
1000
800
600
400
200
0
02/03/20
07/03/20
12/03/20
17/03/20
22/03/20
27/03/20
01/04/20
06/04/20
11/04/20
Date
Per Day Increment (w/o Log) for Delhi
perDayIncrement
200
150
100
50
0
02/03/20
12/03/20
20/03/20
26/03/20
01/04/20
07/04/20
13/04/20
Date
Box Plot for Delhi without Log
perDayIncrement
200
150
100
50
0
Box Plot for Delhi with Log
perDayIncrement
5
4
3
2
1
0
Per Day Increment (Log applied) for Delhi
perDayIncrement
5
4
3
2
1
0
02/03/20
12/03/20
20/03/20
26/03/20
01/04/20
07/04/20
13/04/20
Date


Confirmed cases per day for Rajasthan
Confirmed
Deaths
Cured
800
700
600
500
400
300
200
100
0
03/03/20
08/03/20
15/03/20
18/03/20
23/03/20
28/03/20
02/04/20
07/04/20
12/04/20
Date
Per Day Increment (w/o Log) for Rajasthan
perDayIncrement
140
120
100
80
60
40
20
0
03/03/20
22/03/20
25/03/20
02/04/20
08/04/20
13/04/20
Date
Box Plot for Rajasthan without Log
perDayIncrement
140
120
100
80
60
40
20
0
Box Plot for Rajasthan with Log
perDayIncrement
5
4
3
2
1
0
Per Day Increment (Log applied) for Rajasthan
perDayIncrement
5
4
3
2
1
0
03/03/20
22/03/20
25/03/20
02/04/20
08/04/20
13/04/20
Date


Confirmed cases per day for Uttar Pradesh
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Uttar Pradesh
perDayIncrement
Date
Box Plot for Uttar Pradesh without Log
perDayIncrement
Box Plot for Uttar Pradesh with Log
perDayIncrement
Per Day Increment (Log applied) for Uttar Pradesh
perDayIncrement
Date


Confirmed cases per day for Haryana
Confirmed
Deaths
Cured
175
150
125
100
75
50
25
0
04/03/20
09/03/20
14/03/20
19/03/20
24/03/20
29/03/20
03/04/20
08/04/20
13/04/20
Date
Per Day Increment (w/o Log) for Haryana
perDayIncrement
50
40
30
20
10
0
04/03/20
23/03/20
31/03/20
07/04/20
Date
Box Plot for Haryana without Log
perDayIncrement
50
40
30
20
10
0
Box Plot for Haryana with Log
perDayIncrement
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Per Day Increment (Log applied) for Haryana
perDayIncrement
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
04/03/20
23/03/20
31/03/20
07/04/20
Date


Confirmed cases per day for Ladakh
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Ladakh
perDayIncrement
Date
Box Plot for Ladakh without Log
perDayIncrement
Box Plot for Ladakh with Log
perDayIncrement
Per Day Increment (Log applied) for Ladakh
perDayIncrement
Date

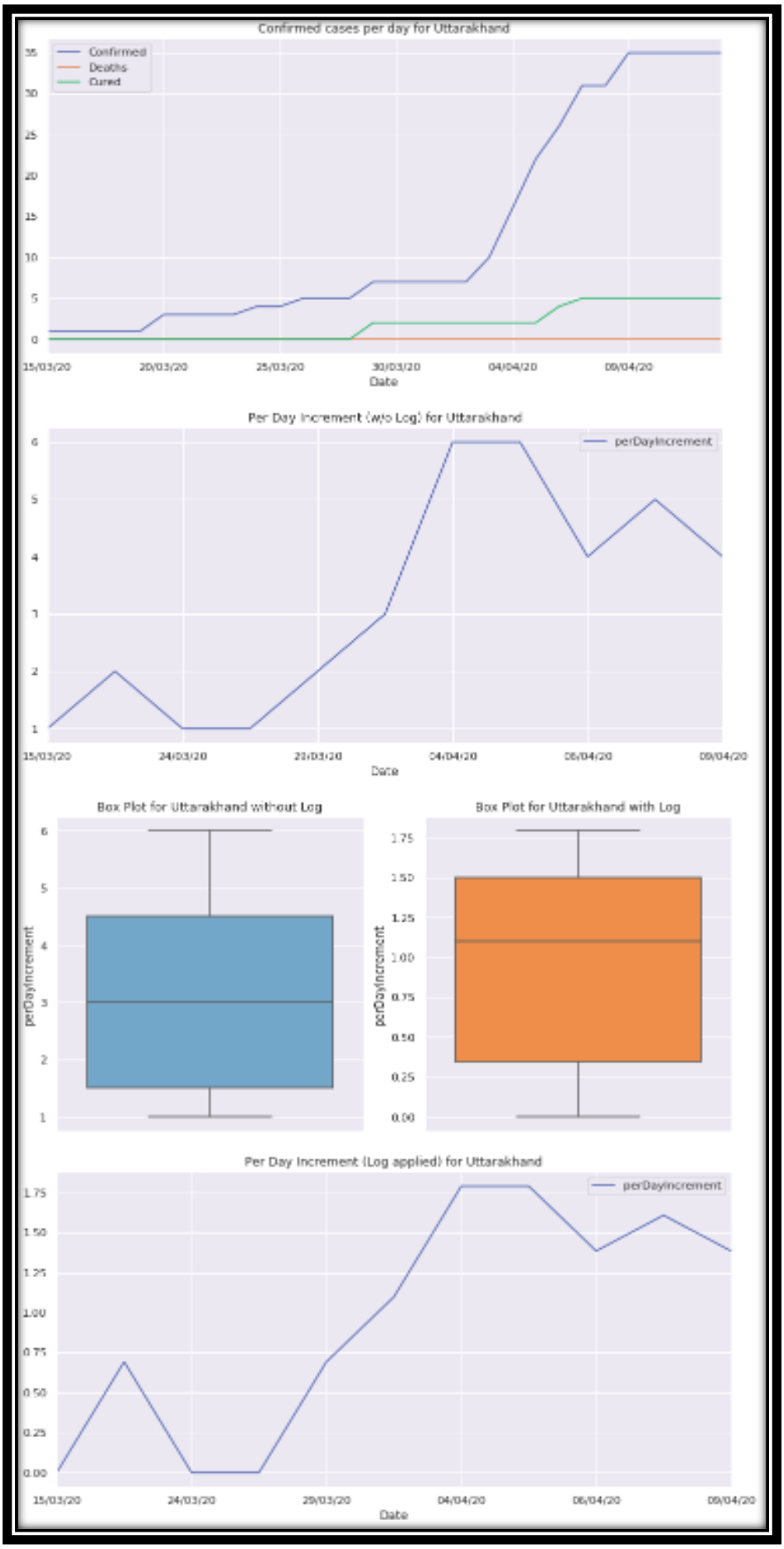

Confirmed cases per day for Tamil Nadu
Confirmed
Deaths
Cured
1000
800
600
400
200
0
07/03/20
12/03/20
17/03/20
22/03/20
27/03/20
01/04/20
06/04/20
11/04/20
Date
Per Day Increment (w/o Log) for Tamil Nadu
perDayIncrement
160
140
120
100
80
60
40
20
0
07/03/20
24/03/20
29/03/20
04/04/20
09/04/20
Date
Box Plot for Tamil Nadu without Log
perDayIncrement
160
140
120
100
80
60
40
20
0
Box Plot for Tamil Nadu with Log
perDayIncrement
5
4
3
2
1
0
Per Day Increment (Log applied) for Tamil Nadu
perDayIncrement
5
4
3
2
1
0
07/03/20
24/03/20
29/03/20
04/04/20
09/04/20
Date


Confirmed cases per day for Karnataka
Confirmed
Deaths
Cured
250
200
150
100
50
0
09/03/20
14/03/20
19/03/20
24/03/20
29/03/20
03/04/20
08/04/20
13/04/20
Date
Per Day Increment (w/o Log) for Karnataka
perDayIncrement
25
20
15
10
5
0
09/03/20
20/03/20
26/03/20
03/04/20
09/04/20
Date
Box Plot for Karnataka without Log
perDayIncrement
25
20
15
10
5
0
Box Plot for Karnataka with Log
perDayIncrement
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Per Day Increment (Log applied) for Karnataka
perDayIncrement
3.0
2.5
2.0
1.5
1.0
0.5
0.0
09/03/20
20/03/20
26/03/20
03/04/20
09/04/20
Date


Confirmed cases per day for Punjab
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Punjab
perDayIncrement
Date
Box Plot for Punjab without Log
perDayIncrement
Box Plot for Punjab with Log
perDayIncrement
Per Day Increment (Log applied) for Punjab
perDayIncrement
Date


Confirmed cases per day for Jammu and Kashmir
Confirmed
Deaths
Cured
250
200
150
100
50
0
09/03/20
14/03/20
19/03/20
24/03/20
29/03/20
03/04/20
08/04/20
13/04/20
Date
Per Day Increment (w/o Log) for Jammu and Kashmir
perDayIncrement
40
30
20
10
0
09/03/20
26/03/20
01/04/20
09/04/20
Date
Box Plot for Jammu and Kashmir without Log
perDayIncrement
40
30
20
10
0
Box Plot for Jammu and Kashmir with Log
perDayIncrement
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Per Day Increment (Log applied) for Jammu and Kashmir
perDayIncrement
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
09/03/20
26/03/20
01/04/20
09/04/20
Date


Confirmed cases per day for Andhra Pradesh
Confirmed
Deaths
Cured
400
300
200
100
0
12/03/20
17/03/20
22/03/20
27/03/20
01/04/20
06/04/20
11/04/20
Date
Per Day Increment (w/o Log) for Andhra Pradesh
perDayIncrement
50
40
30
20
10
0
12/03/20
24/03/20
29/03/20
03/04/20
08/04/20
Date
Box Plot for Andhra Pradesh without Log
perDayIncrement
50
40
30
20
10
0
Box Plot for Andhra Pradesh with Log
perDayIncrement
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Per Day Increment (Log applied) for Andhra Pradesh
perDayIncrement
4.0
3.5
1.5
1.0
0.5
0.0
12/03/20
24/03/20
29/03/20
03/04/20
08/04/20
Date

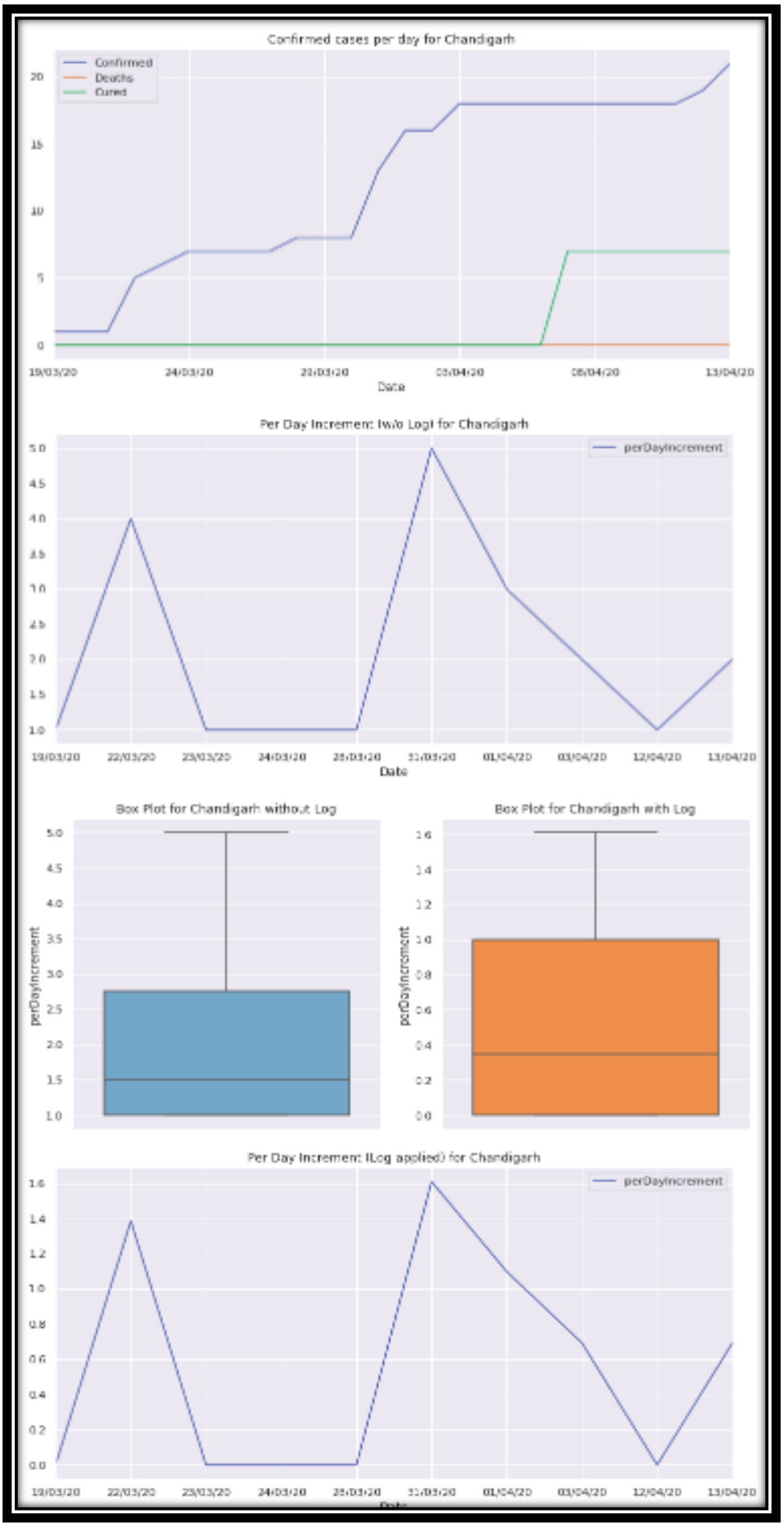

Confirmed cases per day for Uttarakhand
Confirmed
Deaths
Cured
15/03/20
20/03/20
25/03/20
30/03/20
04/04/20
09/04/20
Date
Per Day Increment (w/o Log) for Uttarakhand
perDayIncrement
15/03/20
24/03/20
29/03/20
04/04/20
09/04/20
Date
Box Plot for Uttarakhand without Log
perDayIncrement
Box Plot for Uttarakhand with Log
perDayIncrement
Per Day Increment (Log applied) for Uttarakhand
perDayIncrement
15/03/20
24/03/20
29/03/20
04/04/20
09/04/20
Date

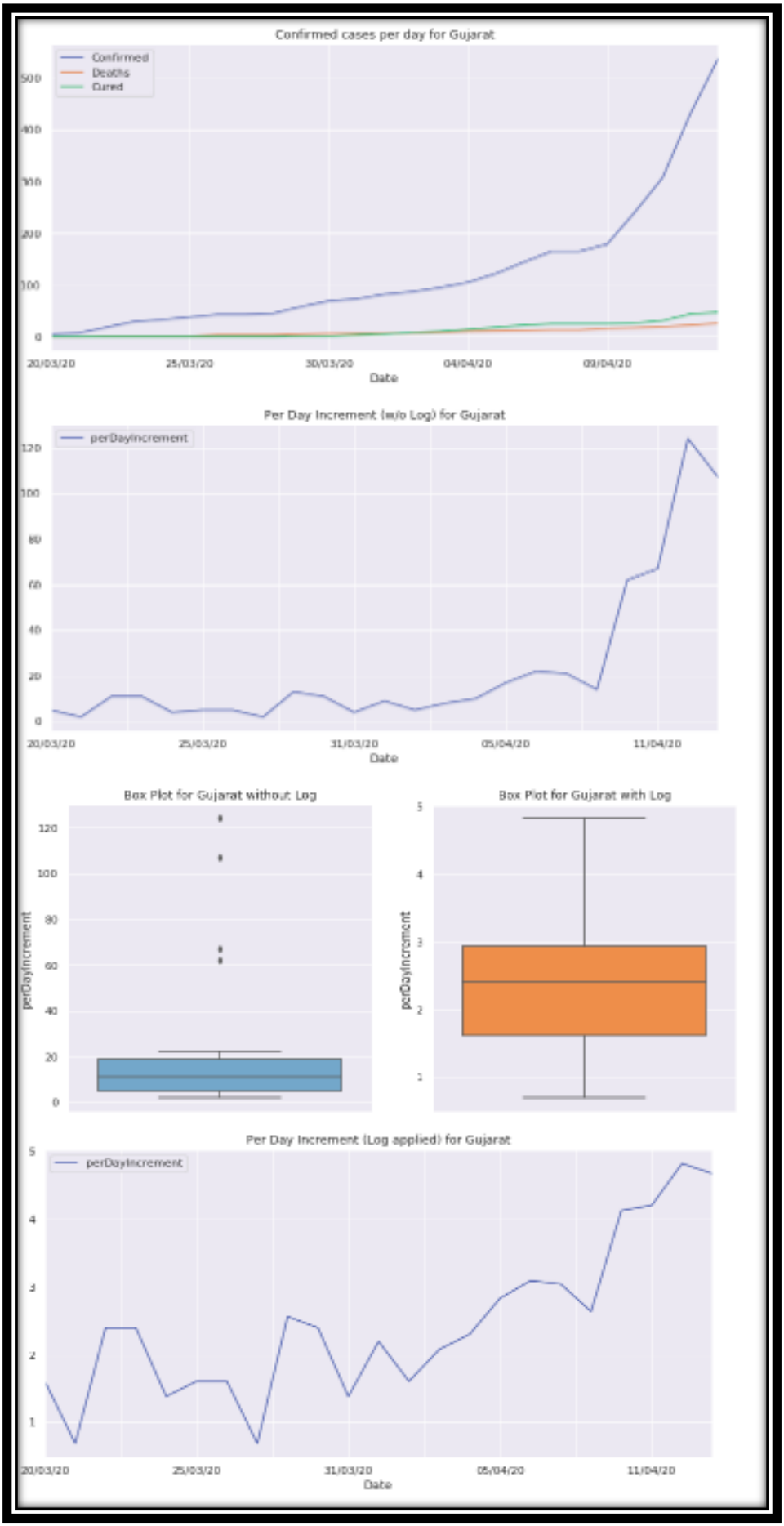

Confirmed cases per day for Odisha
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Odisha
perDayIncrement
Date
Box Plot for Odisha without Log
perDayIncrement
Box Plot for Odisha with Log
perDayIncrement
Per Day Increment (Log applied) for Odisha
perDayIncrement
Date


Confirmed cases per day for Puducherry
Confirmed
Deaths
Cured
7
6
5
4
3
2
1
0
18/03/20
25/03/20
28/03/20
02/04/20
07/04/20
12/04/20
Date
Per Day Increment (w/o Log) for Puducherry
perDayIncrement
2.0
1.8
1.6
1.4
1.2
1.0
18/03/20
01/04/20
03/04/20
11/04/20
Date
Box Plot for Puducherry without Log
perDayIncrement
2.0
1.8
1.6
1.4
1.2
1.0
Box Plot for Puducherry with Log
perDayIncrement
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Per Day Increment (Log applied) for Puducherry
perDayIncrement
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
18/03/20
01/04/20
03/04/20
11/04/20
Date


Confirmed cases per day for West Bengal
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for West Bengal
perDayIncrement
Date
Box Plot for West Bengal without Log
perDayIncrement
Box Plot for West Bengal with Log
perDayIncrement
Per Day Increment (Log applied) for West Bengal
perDayIncrement
Date


Confirmed cases per day for Chhatisgarh
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Chhattisgarh
perDayIncrement
Date
Box Plot for Chhattisgarh without Log
perDayIncrement
Box Plot for Chhattisgarh with Log
perDayIncrement
Per Day Increment (Log applied) for Chhattisgarh
perDayIncrement
Date

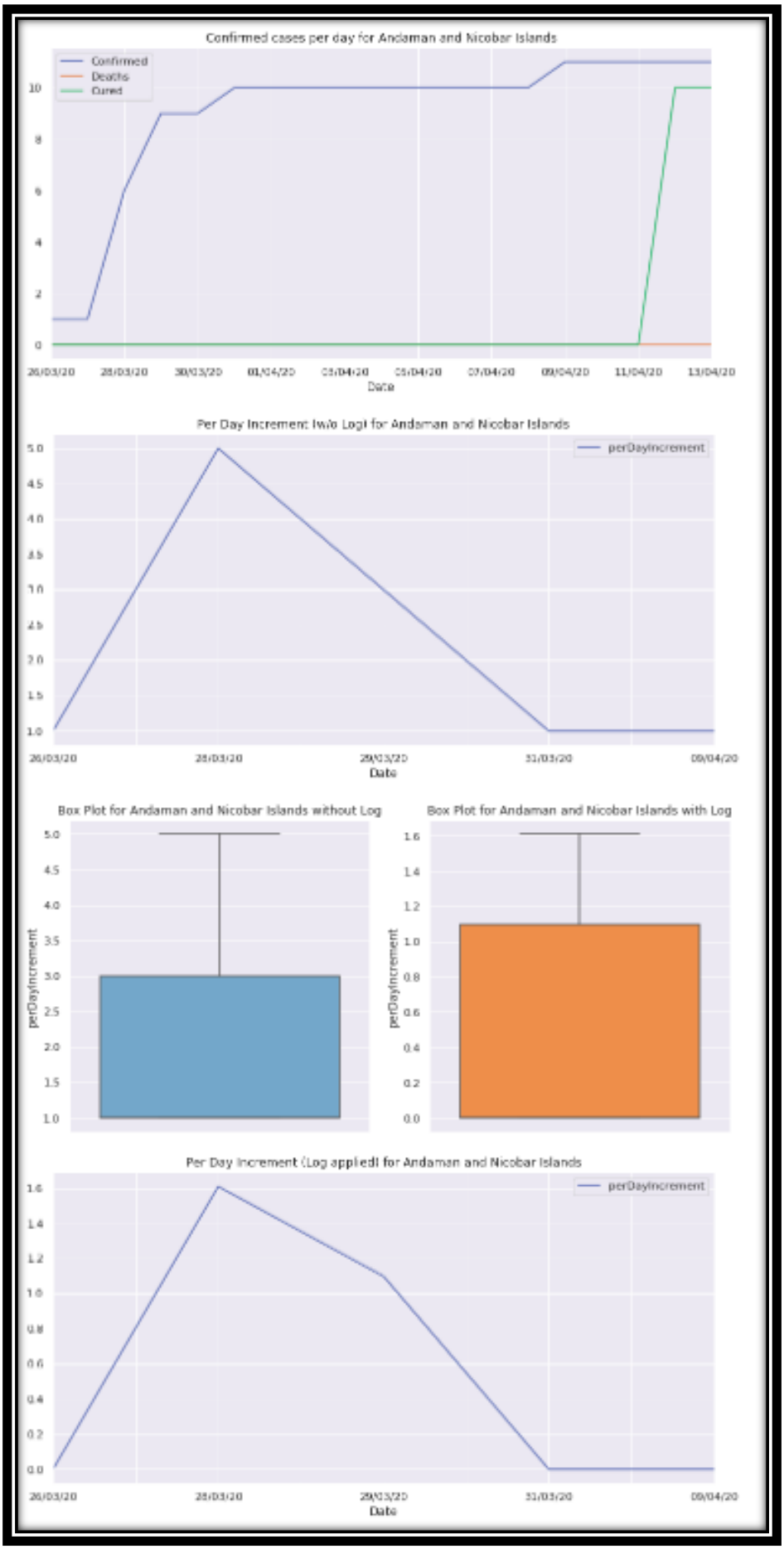

Confirmed cases per day for Chandigarh
Confirmed
Deaths
Cured
Date
Per Day Increment: Iw/o Log) for Chandigarh
perDayIncrement
Date
Box Plot for Chandigarh without Log
Box Plot for Chandigarh with Log
perDayIncrement
Per Day Increment: ILog applied) for Chandigarh
perDayIncrement

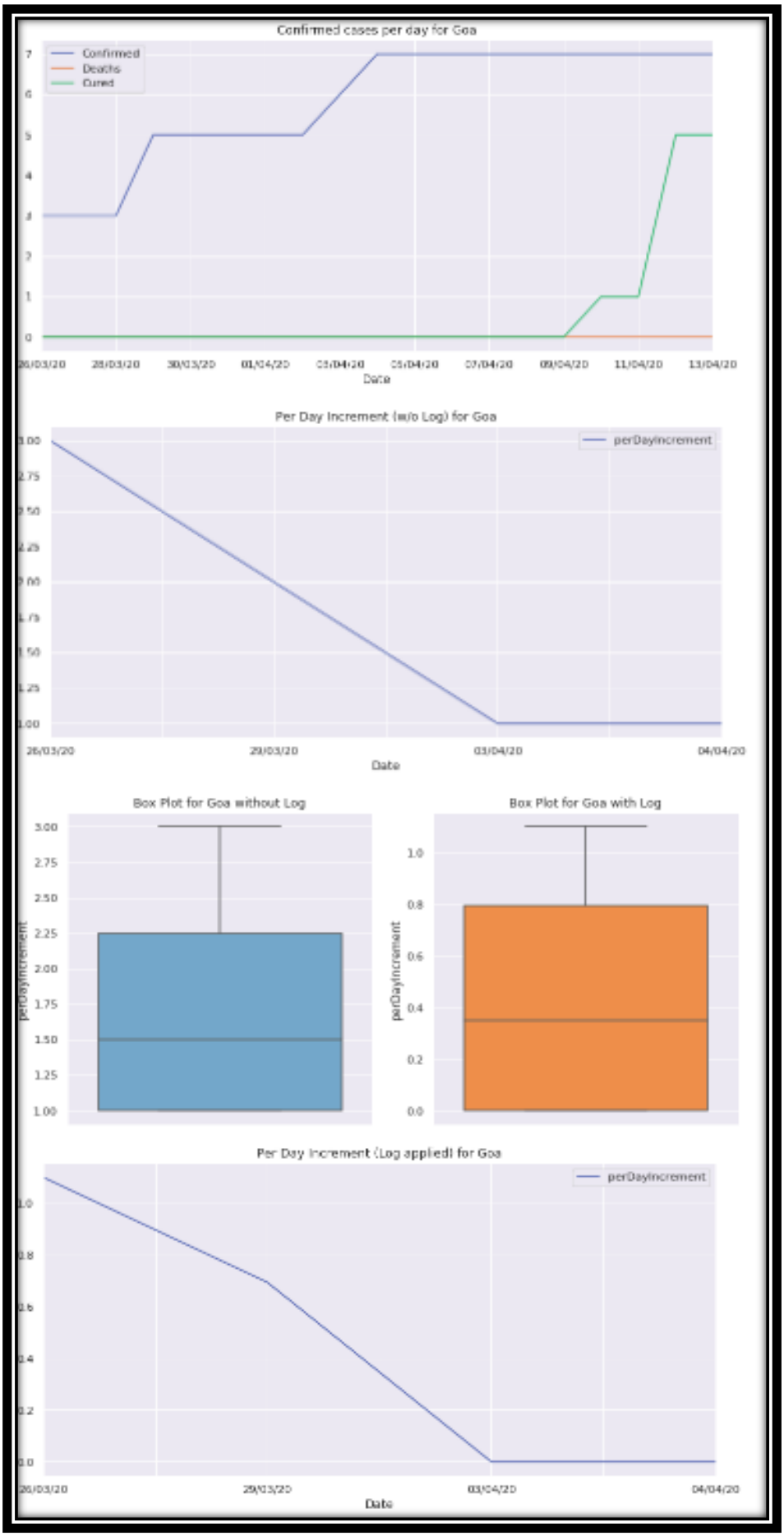

Confirmed cases per day for Gujarat
Confirmed
Deaths
Cured
500
400
300
200
100
0
20/03/20
25/03/20
30/03/20
04/04/20
09/04/20
Date
Per Day Increment (w/o Log) for Gujarat
perDayIncrement
120
100
80
60
40
20
0
20/03/20
25/03/20
31/03/20
05/04/20
11/04/20
Date
Box Plot for Gujarat without Log
perDayIncrement
120
100
80
60
40
20
0
Box Plot for Gujarat with Log
perDayIncrement
5
4
3
2
1
Per Day Increment (Log applied) for Gujarat
perDayIncrement
5
4
3
2
1
20/03/20
25/03/20
31/03/20
05/04/20
11/04/20
Date


Confirmed cases per day for Himachal Pradesh
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Himachal Pradesh
perDayIncrement
Date
Box Plot for Himachal Pradesh without Log
perDayIncrement
Box Plot for Himachal Pradesh with Log
perDayIncrement
Per Day Increment (Log applied) for Himachal Pradesh
perDayIncrement
Date

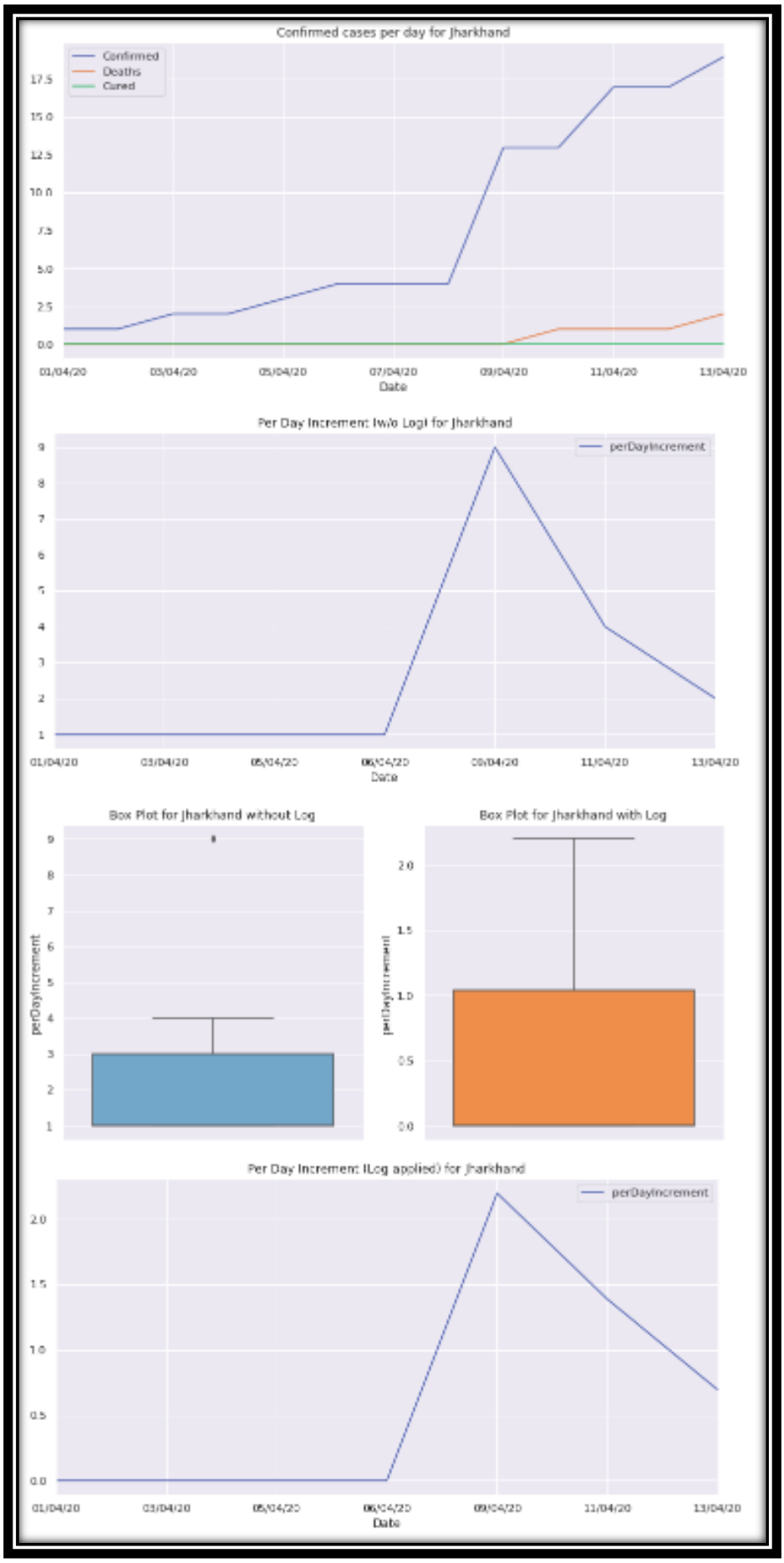

Confirmed cases per day for Madhya Pradesh
Confirmed
Deaths
Cured
600
500
400
300
200
100
0
21/03/20
26/03/20
31/03/20
05/04/20
10/04/20
Date
Per Day Increment (w/o Log) for Madhya Pradesh
perDayIncrement
175
150
125
100
75
50
25
0
21/03/20
24/03/20
26/03/20
30/03/20
02/04/20
05/04/20
09/04/20
12/04/20
Date
Box Plot for Madhya Pradesh without Log
perDayIncrement
175
150
125
100
75
50
25
0
Box Plot for Madhya Pradesh with Log
perDayIncrement
5
4
3
2
1
0
Per Day Increment (Log applied) for Madhya Pradesh
perDayIncrement
5
4
3
2
1
0
21/03/20
24/03/20
26/03/20
30/03/20
02/04/20
05/04/20
09/04/20
12/04/20
Date


Confirmed cases per day for Bihar
Confirmed
Deaths
Cured
60
50
40
30
20
10
0
22/03/20
27/03/20
01/04/20
06/04/20
11/04/20
Date
Per Day Increment Iwlo Logl for Bihar
perDayIncrement
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
22/03/20
25/03/20
28/03/20
30/03/20
02/04/20
04/04/20
08/04/20
10/04/20
Date
Box Plot for Bihar without Log
perDayIncrement
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
Box Plot for Bihar with Log
perDayIncrement
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Per Day Increment (Log applied) for Bihar
perDayIncrement
3.0
2.5
2.0
1.5
1.0
0.5
0.0
22/03/20
25/03/20
28/03/20
30/03/20
02/04/20
04/04/20
08/04/20
10/04/20
Date

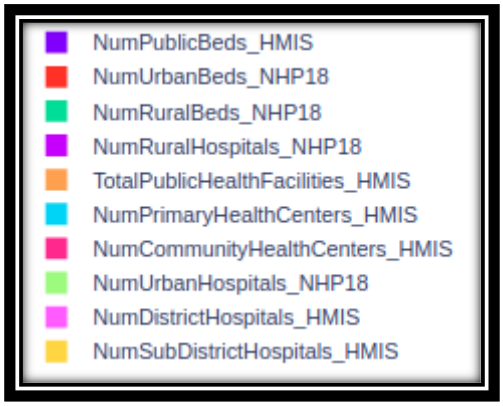

Confirmed cases per day for Andaman and Nicobar Islands
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Andaman and Nicobar Islands
perDayIncrement
Date
Box Plot for Andaman and Nicobar Islands without Log
Box Plot for Andaman and Nicobar Islands with Log
perDayIncrement
Per Day Increment (Log applied) for Andaman and Nicobar Islands
perDayIncrement
Date


Confirmed cases per day for Goa
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Goa
perDayIncrement
Date
Box Plot for Goa without Log
perDayIncrement
Box Plot for Goa with Log
perDayIncrement
Per Day Increment (Log applied) for Goa
perDayIncrement
Date

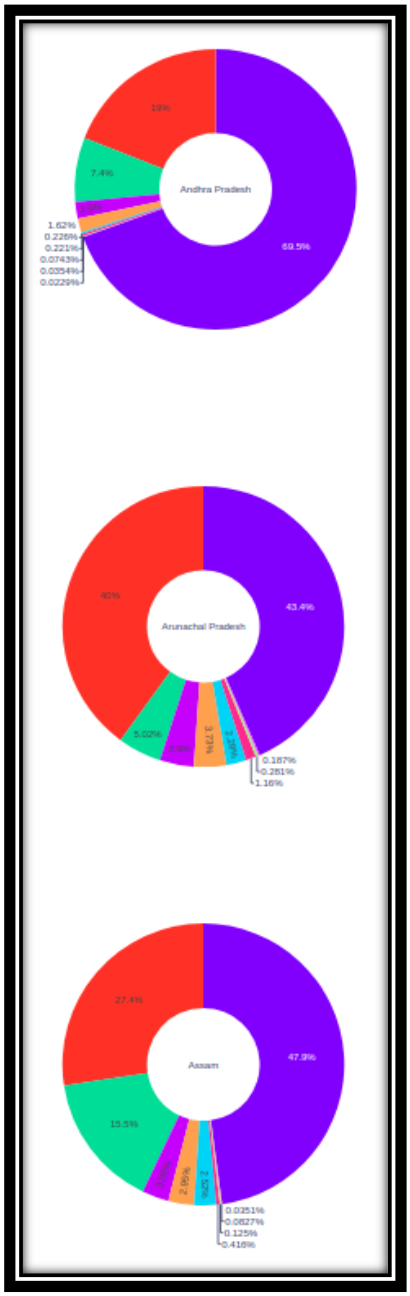

Confirmed cases per day for Assam
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Assam
perDayIncrement
Date
Box Plot for Assam without Log
perDayIncrement
Box Plot for Assam with Log
perDayIncrement
Per Day Increment (Log applied) for Assam
perDayIncrement
Date

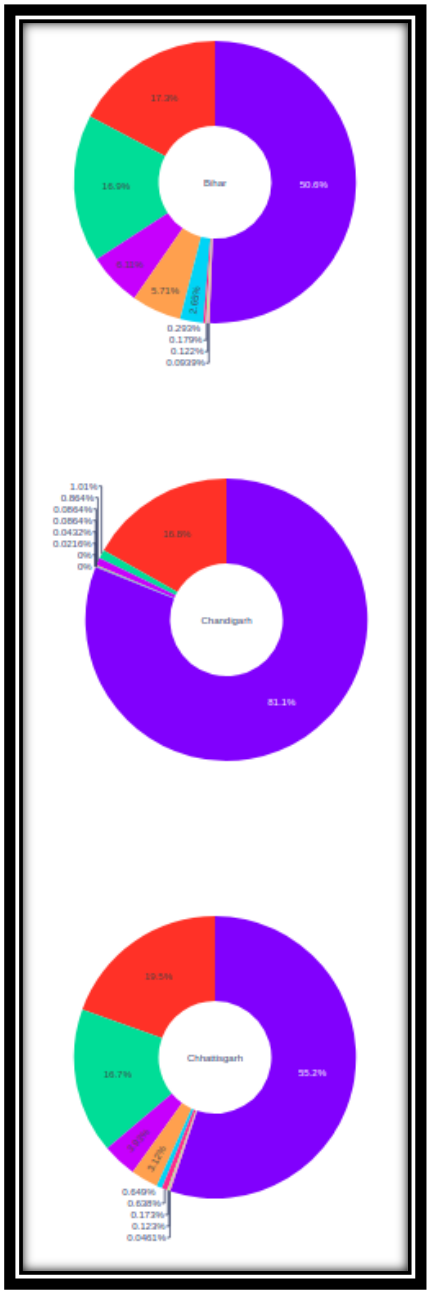

Confirmed cases per day for Jharkhand
Confirmed
Deaths
Cured
Date
Per Day Increment (w/o Log) for Jharkhand
perDayIncrement
Date
Box Plot for Jharkhand without Log
perDayIncrement
Box Plot for Jharkhand with Log
perDayIncrement
Per Day Increment (Log applied) for Jharkhand
perDayIncrement
Date

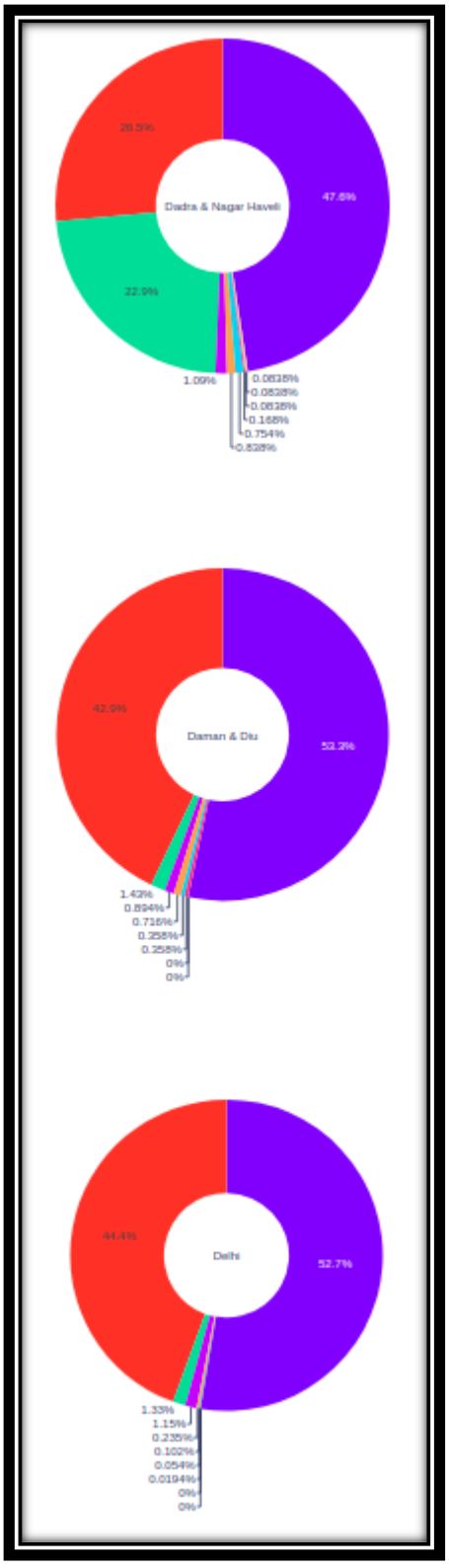
PIE CHARTS FOR HOSPITALS IN EACH STATE

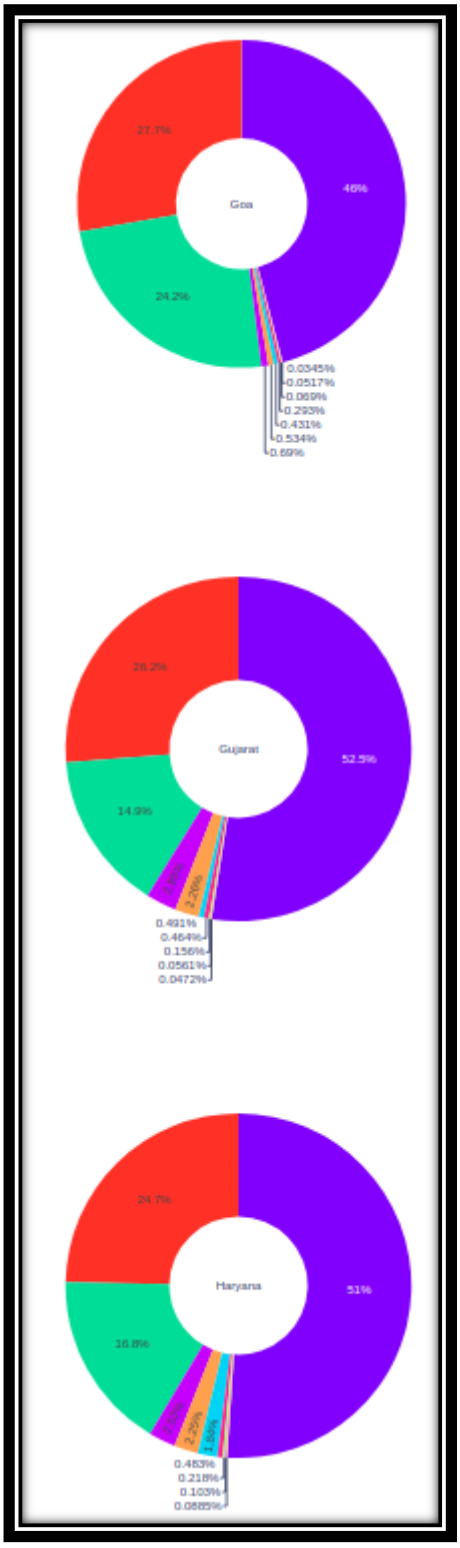


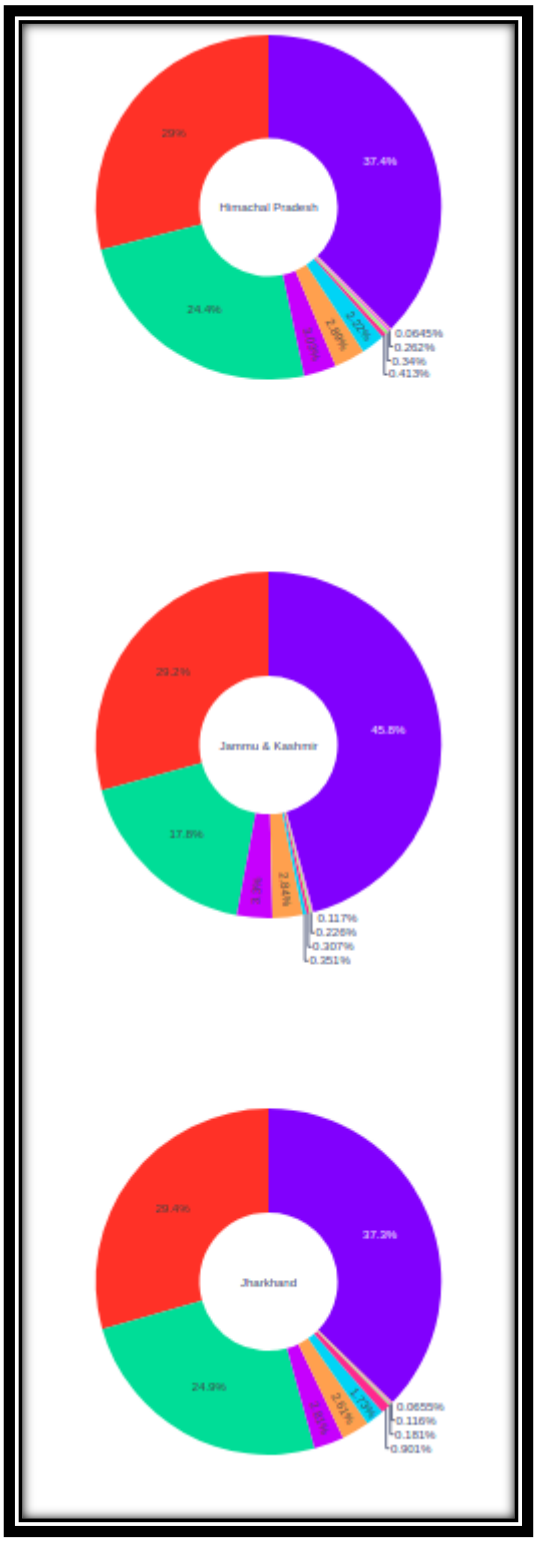
KEY BINDINGS

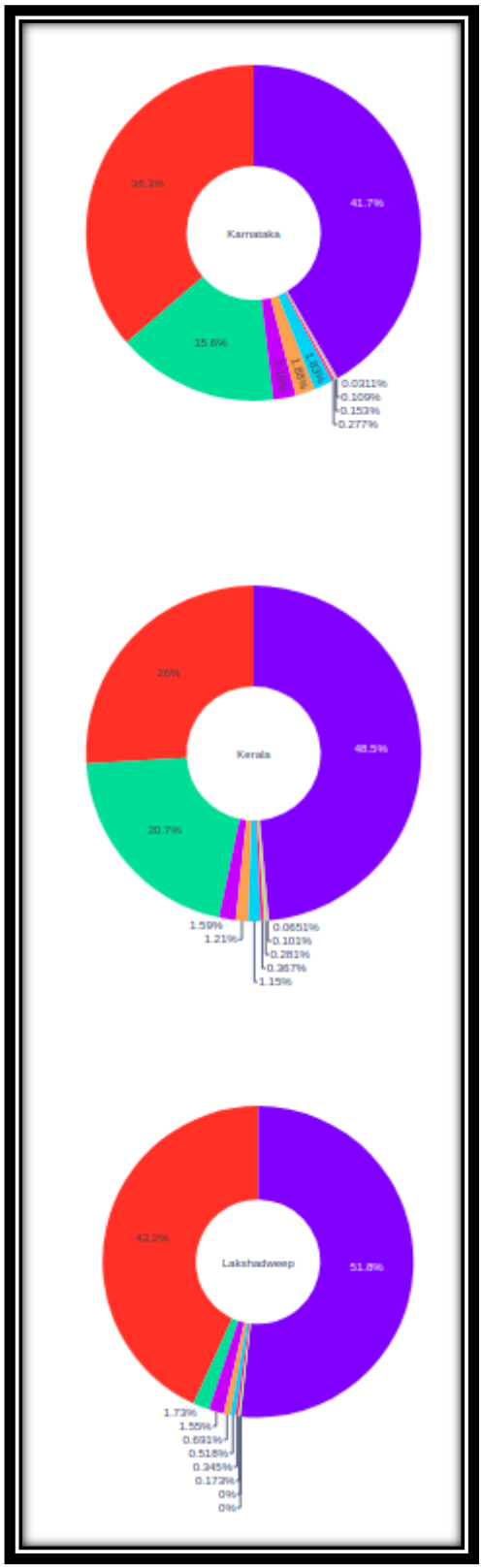


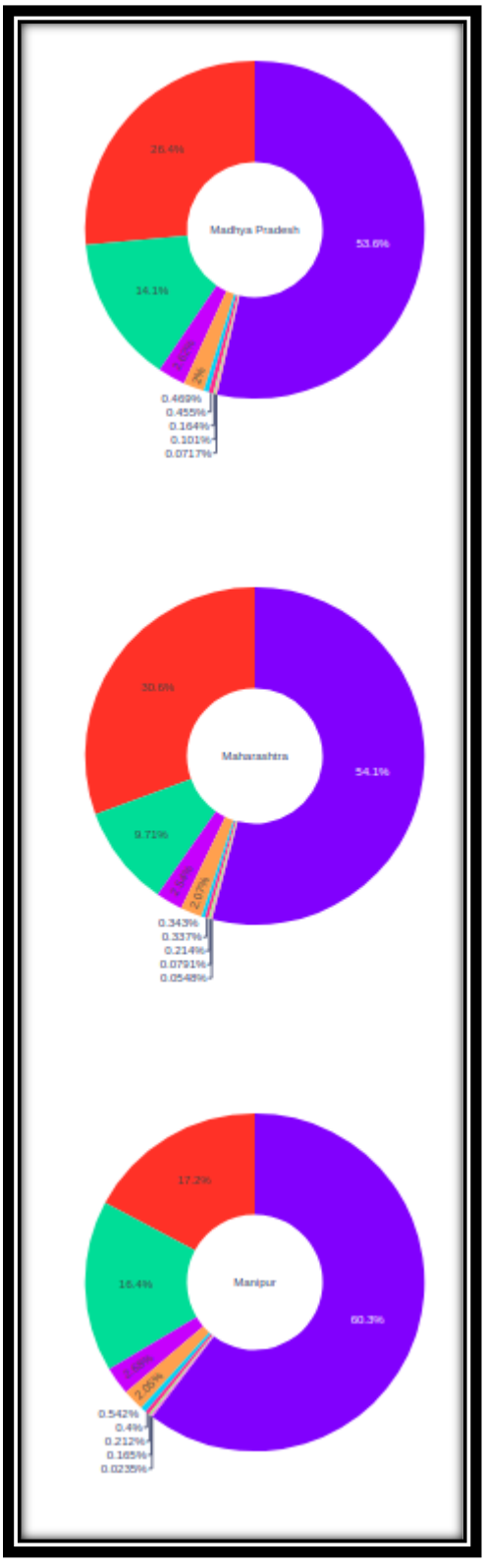


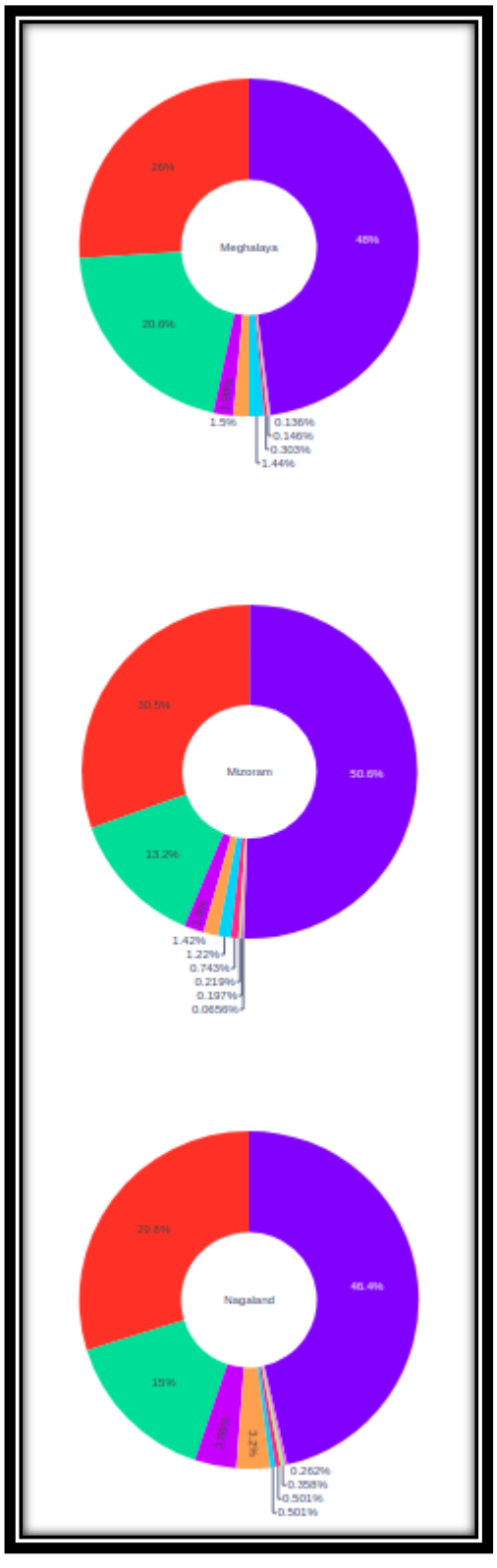

Dadra & Nagar Haveli
47.8%
26.5%
22.9%
1.09%
0.0838%
0.0838%
0.0838%
0.168%
0.754%
0.838%
Daman & Diu
52.2%
42.9%
1.42%
0.894%
0.716%
0.358%
0.358%
0%
0%
Delhi
52.7%
44.4%
1.33%
1.15%
0.235%
0.102%
0.054%
0.0194%
0%
0%

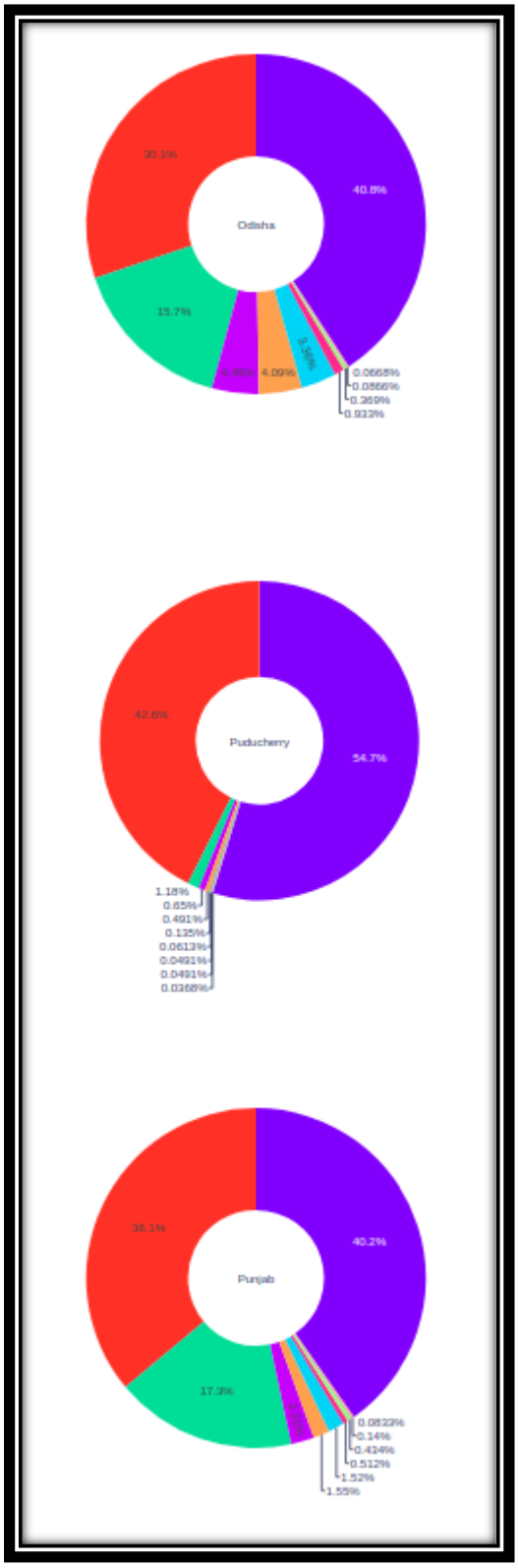

Goa
46%
27.7%
24.2%
0.0345%
0.0517%
0.069%
0.293%
0.431%
0.534%
0.69%
Gujarat
52.5%
26.2%
14.9%
2.26%
0.491%
0.464%
0.156%
0.0561%
0.0472%
Haryana
51%
24.7%
16.8%
2.25%
1.66%
0.483%
0.218%
0.103%
0.0885%

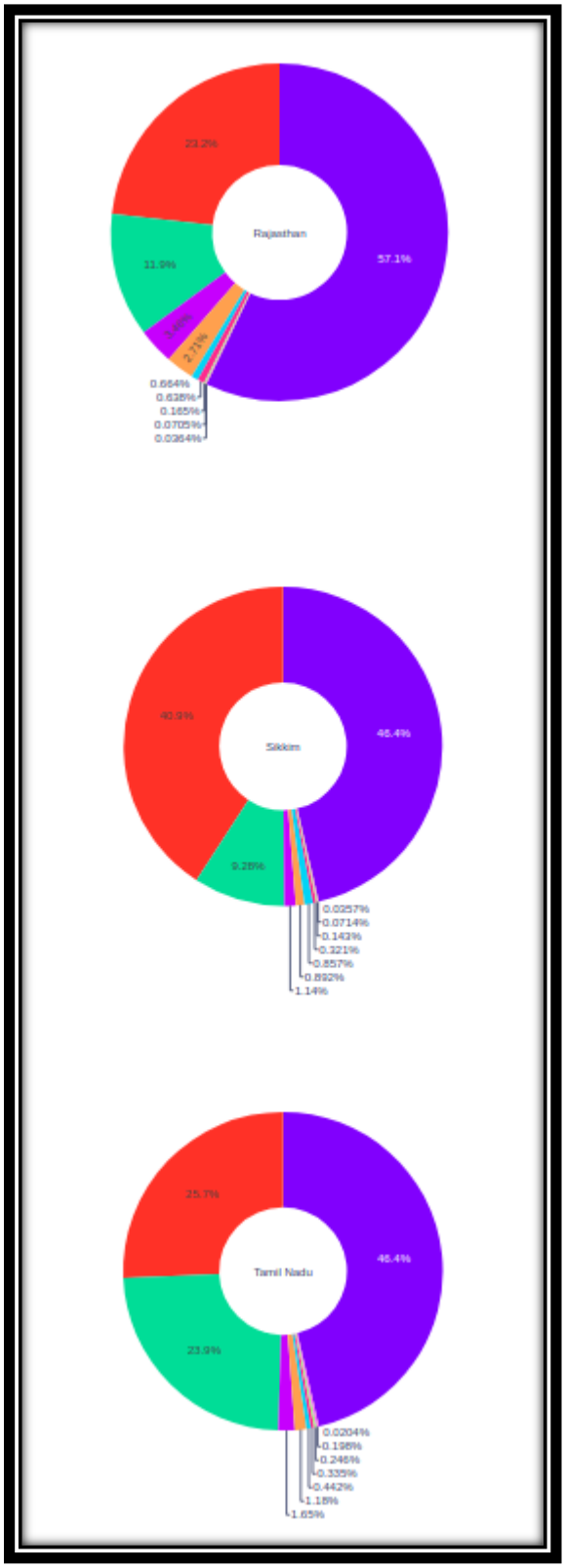

Himachal Pradesh
37.4%
29%
24.4%
0.0645%
0.262%
0.34%
0.413%
Jammu & Kashmir
45.8%
29.2%
17.8%
0.117%
0.226%
0.307%
0.351%
Jharkhand
37.2%
29.4%
24.9%
0.0655%
0.116%
0.181%
0.901%

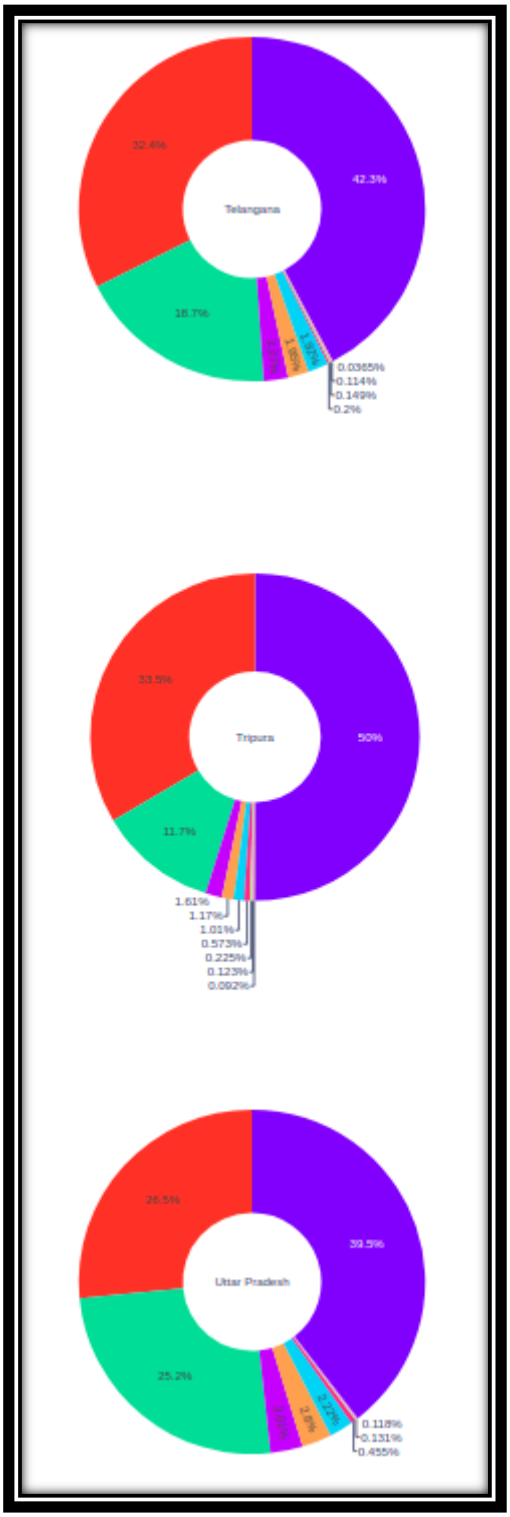

Karnataka
41.7%
15.6%
0.0211%
0.109%
0.153%
0.277%
Kerala
48.5%
20.7%
1.59%
1.21%
0.0651%
0.101%
0.261%
0.367%
1.15%
Lakshadweep
51.8%
1.72%
1.55%
0.691%
0.518%
0.345%
0.172%
0%
0%

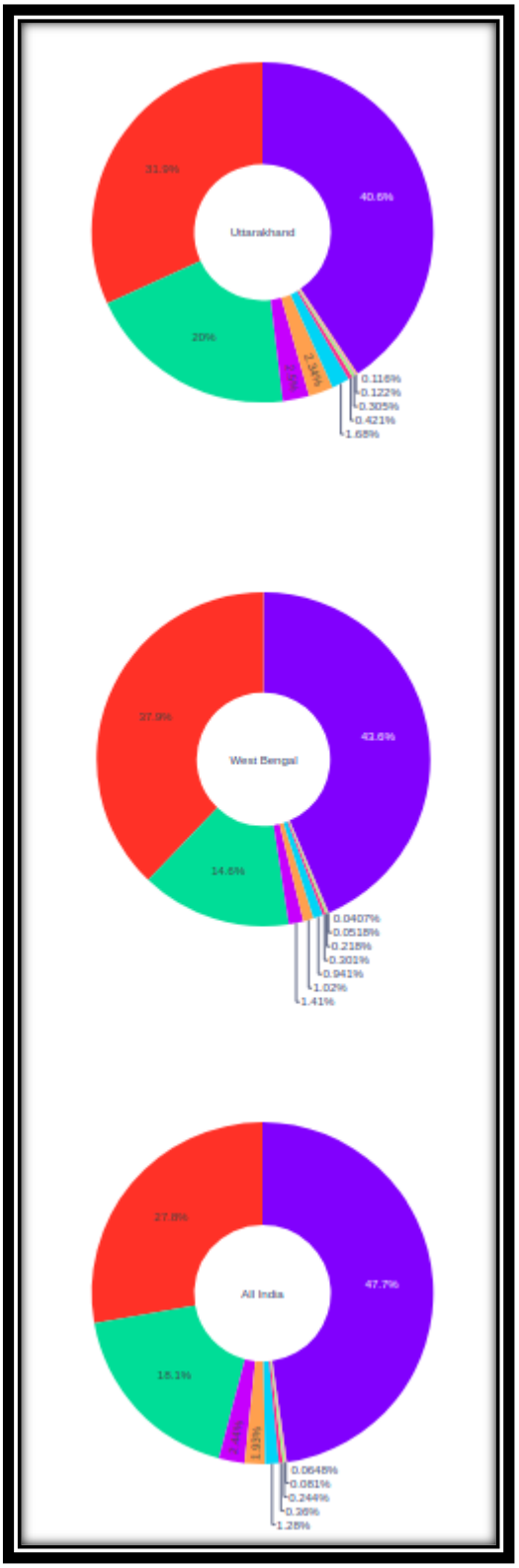

Madhya Pradesh
53.6%
26.4%
14.1%
0.469%
0.455%
0.164%
0.101%
0.0717%
Maharashtra
54.1%
30.6%
9.71%
2.07%
0.343%
0.337%
0.214%
0.0791%
0.0548%
Manipur
60.3%
17.2%
16.4%
2.05%
0.542%
0.4%
0.212%
0.165%
0.0235%


Meghalaya
45%
26%
20.6%
1.5%
0.136%
0.146%
0.303%
1.44%
Mizoram
50.6%
30.5%
13.2%
1.42%
1.22%
0.743%
0.219%
0.197%
0.0659%
Nagaland
46.4%
29.8%
15%
3.13%
0.262%
0.358%
0.501%
0.501%


Odisha
40.8%
30.1%
15.7%
4.09%
0.0665%
0.0666%
0.369%
0.932%
Puducherry
54.7%
42.6%
1.18%
0.65%
0.491%
0.135%
0.0613%
0.0491%
0.0491%
0.0368%
Punjab
40.2%
36.1%
17.2%
0.0822%
0.14%
0.434%
0.512%
1.52%
1.55%


Rajasthan
57.1%
23.2%
11.9%
2.71%
0.664%
0.638%
0.165%
0.0705%
0.0364%
Sikkim
46.4%
40.9%
9.28%
0.0257%
0.0714%
0.143%
0.321%
0.857%
0.892%
1.14%
Tamil Nadu
46.4%
25.3%
23.9%
0.0204%
0.198%
0.246%
0.335%
0.442%
1.18%
1.65%


Telangana
42.3%
32.4%
18.7%
0.0365%
0.114%
0.149%
0.2%
Tripura
50%
33.5%
11.7%
1.61%
1.17%
1.01%
0.573%
0.225%
0.123%
0.092%
Uttar Pradesh
39.5%
26.5%
25.2%
0.118%
0.131%
0.455%


Uttarakhand
40.6%
31.9%
20%
0.116%
0.122%
0.305%
0.421%
1.68%
West Bengal
42.6%
27.9%
14.6%
0.0407%
0.0518%
0.218%
0.301%
0.941%
1.02%
1.41%
All India
47.7%
27.8%
18.1%
0.0648%
0.081%
0.244%
0.36%
1.28%