Dissertation on

"Probabilistic Models for Predicting COVID-19 Pandemic Spread"

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

Bachelor of Technology
in
Computer Science & Engineering

UE17CS490B – Capstone Project Phase - 2

Submitted by:

K Suhas	**PES1201700816**
Kethan M V	**PES1201701085**
Venkatesh K	**PES1201701626**
Akash P S	**PES1201701773**

Under the guidance of

Prof. Nitin V Pujari
Dean, IQAC PES
University

January - May 2021

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
FACULTY OF ENGINEERING
PES UNIVERSITY
(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

PES UNIVERSITY

(Established under Karnataka Act No. 16 of 2013)
100ft Ring Road, Bengaluru – 560 085, Karnataka, India

FACULTY OF ENGINEERING

CERTIFICATE

This is to certify that the dissertation entitled

'Probabilistic Models for Predicting COVID-19 Pandemic Spread'

is a bonafide work carried out by

K Suhas	**PES1201700816**
Kethan M V	**PES1201701085**
Venkatesh K	**PES1201701626**
Akash P S	**PES1201701773**

in partial fulfilment for the completion of eighth semester Capstone Project Phase - 2 (UE17CS490B) in the Program of Study - Bachelor of Technology in Computer Science and Engineering under rules and regulations of PES University, Bengaluru during the period Jan. 2021 – May. 2021. It is certified that all corrections / suggestions indicated for internal assessment have been incorporated in the report. The dissertation has been approved as it satisfies the 8th semester academic requirements in respect of project work.

Signature **Prof. Nitin V Pujari** Dean IQAC	Signature Dr. Shylaja S S Chairperson	Signature Dr. B K Keshavan Dean of Faculty

External Viva

Name of the Examiners **Signature with Date**

1. _____ _____

2. _____**1.** ____ _____

DECLARATION

We hereby declare that the Capstone Project Phase - 2 entitled **"Probabilistic Models for Predicting COVID-19 Pandemic Spread"** has been carried out by us under the guidance of Prof. Nitin V Pujari, Dean IQAC and submitted in partial fulfilment of the course requirements for the award of degree of **Bachelor of Technology** in **Computer Science and Engineering** of **PES University, Bengaluru** during the academic semester January – May 2021. The matter embodied in this report has not been submitted to any other university or institution for the award of any degree.

PES1201700816	**K Suhas**	
PES1201701085	**Kethan M V**	
PES1201701626	**Venkatesh K**	
PES1201701773	**Akash P S**	

ACKNOWLEDGEMENT

I would like to express my gratitude to Prof. Nitin V Pujari, Department of Computer Science and Engineering, PES University, for his continuous guidance, assistance, and encouragement throughout the development of this UE17CS490B - Capstone Project Phase – 2.

I am grateful to the project coordinators Prof. Silviya Nancy J, Prof. Sunitha R for organizing, managing, and helping with the entire process.

I take this opportunity to thank Dr. Shylaja S S, Chairperson, Department of Computer Science and Engineering, PES University, for all the knowledge and support I have received from the department. I would like to thank Dr. B.K. Keshavan, Dean of Faculty, PES University for his help.

I am deeply grateful to Dr. M. R. Doreswamy, Chancellor, PES University, Prof. Jawahar Doreswamy, Pro Chancellor – PES University, Dr. Suryaprasad J, Vice-Chancellor, PES University for providing to me various opportunities and enlightenment every step of the way. Finally, this project could not have been completed without the continual support and encouragement I have received from my family and friends.

ABSTRACT

The world is currently going through the onslaught of COVID-19 pandemic resulting in lockdown and other associated direct and indirect effects. Coronavirus disease 2019 (COVID-19) is one which causes unprecedented spread when an entity called human being comes in contact with each other. The virus belongs to the family of Coronaviridae which causes health related problems causing severe acute respiratory illness in human beings coming in contact with each other. The world has placed epidemic modelling at the forefront to enable the world and its stakeholders to take informed decision making in containing the spread. In this work we will employ various probabilistic models to predict the spread of COVID-19 virus and validate their efficacy by testing it with actual data. We take advantage of the large amount of data to predict the COVID-19 pandemic spread. The results obtained from various probabilistic models will be compared based on the relevant metrics and hence enable us to obtain justifiable and verifiable conclusions.

TABLE OF CONTENTS

CHAPTER 1

INTRODUCTION

On November 17th 2019, a new type of virus named "SARS COV-2" (Severe Acute Respiratory Syndrome Coronavirus 2) originated in Hubei province capital Wuhan, China. This belongs to β Coronavirus of Zoonotic origin among α, β, γ, Δ. The World Health Organization (WHO) announced that coronavirus disease (COVID-19) has spread throughout the world and has been declared as a pandemic on March 11th 2020. The virus has spread widely, and the number of cases rising daily with numbers reaching 148,506,439 as on 27th April 2021 at 11:55 AM IST in the world. All the respective government are working towards slowing down the spread by asking and imposing safety measures to prevent the spread. Pharmaceutical interventions such as vaccination and antiviral drugs have recently become available around the world. However, the onslaught of the pandemic is far from over which is especially true for India as it has declared a second wave of the disease spread with daily cases reaching up to 3,52,991 on 25th April 2021. Addressing the COVID-19 disease will mainly depend on the effective implementation of public health measures such as social distancing, disease contact tracing, quarantine and so on till more doses of vaccination is effectively administered to curb the spread of the virus.

The world has placed epidemic modelling at the forefront to enable the world and its stakeholders to take informed decision making in containing the spread. Nonetheless, modelling and predicting the spread of COVID-19 remain a challenge mainly due to the uncertainty present in the spread of the disease and limitation in availability of appropriate dataset, which is a challenge to generate from the raw data.

We now have substantially large amount of raw data as well as the research outcomes obtained by scientists and researchers across the globe which has motivated us to utilize the latest open-source data ecosystem and contribute probabilistic models for predicting

COVID-19. Here we make use of these raw datasets and perform data pre-processing to create different data frames suitable to our problem statement and develop various appropriate probabilistic models for predicting COVID-19 spread and the same will be verified against the standard and derived metrics using the open-source raw dataset.

CHAPTER 2

PROBLEM STATEMENT

Prediction models so far have different sources of uncertainty like parameter uncertainty, uncertainty present in data, variation in different types of data or model type used and they highlight the challenges in modelling and predicting the future spread of the pandemic with only limited data.

Traditional approaches for modelling COVID-19 are based on deterministic models that often rely on average data, and thus only provide expected results. These models do not propagate the variability and uncertainty of data.

Our solution is to develop probabilistic models for predicting COVID-19 spread. Probabilistic models are statistical models that incorporates random variables into the model to account for the uncertainty in the data. These models take into account of the fact that we rarely know anything about the situation. Some of the commonly used models include regression models, Markov chain Monte Carlo simulations and tree-based models.

The performance of probabilistic models will heavily depend on the input data. Hence, data pre-processing is one of the most important steps to ensure that the data representation can support our probabilistic models.

In this work, we will analyze different types of data available on COVID-19, understand the data and make a decision to create the most suitable dataset for our problem. We will also contribute probabilistic models for predicting COVID-19 spread and compare the predictive performances of all the models developed so we can have a greater understanding of the virus's spread and which model will be best suited for the problem statement.

CHAPTER 3

LITERATURE SURVEY

Here, we have presented the current knowledge of the area that will help in our study.

3.1 Understanding COVID-19 transmission through Bayesian probabilistic modeling and GIS-based Voronoi approach: a policy perspective [1]

Hemant Bherwani · Saima Anjum · Suman Kumar · Sneha Gautam Ankit Gupta · Himanshu Kumbhare · Avneesh Anshul · Rakesh Kumar.
Accepted: 1 July 2020
Indexing: Springer.

In this paper, the authors identify the relation between the population density and the transmission of COVID-19 when the lockdown was implemented in India using Bayesian Probabilistic Modeling and also with the above correlation, they constructed TP/Voronoi diagrams for a few states to show the results graphically.

Methodology: Bayesian Change Point Analysis is carried out for 8 states (Maharashtra, Tamil Nadu, Uttar Pradesh, Madya Pradesh, Delhi, Gujrat, West Bengal and Rajasthan) in India to find the COVID-19 infected cases in each state. Delta is calculated by considering the first case and the date when the lockdown was implemented. Delta shows the immediate action taken by the respective state government in containing the spread of the virus. Cases per population, Cases per unit population Density, Cases per unit area are calculated and are compared with Delta using Pearson's Correlation.

The result is verified by performing t-test between the parameters. Voronoi/TP Diagrams are constructed for the states with the parameters (Cases, Population) in order to show the correlation graphically.

Conclusion:

I. There is a strong correlation between population density and the COVID-19 cases confirmed which is proven by t-test with 95% confidence interval.
II. Bayesian Probabilistic Models is used to know the spread of COVID-19 cases in all the states of India and also in other countries.
III. It also indicates the effects of delayed action by the states from the date of detecting the first case.
IV. With the help of Voronoi/TP diagrams we can find the hotspots and safe zones in the country. It also helps in ranking the states according to the level of infection which subsequently can be used in phased unlocking of the states.

3.2 The challenges of modeling and forecasting the spread of COVID-19 [2]

Andrea L. Bertozzi, Elisa Franco, George Mohler, Martin B. Short, and Daniel Sledge
Accepted: 2 July 2020
Indexing: Pnas.org

In this paper, the author highlights the challenges faced in modelling and predicting the spread of COVID-19. Parsimonious models were used to show how these models are connected the each other and they also highlight the dangers of relaxing public health interventions when vaccines and antiviral drugs are absent.

Methodology: Presented three models for transmission rate.

a. Exponential Behavior (During the first stage)
b. Branching Process (During the analysis stage)
c. SIR (Susceptible – Infected – Resistant) Compartment model (During the peak of the disease spread)

Results:

V. The reproductive number varies by both location and time and is increased based on social distancing measures.
VI. Fatality rate data and confirmed case data will vary by both time and location.
VII. Public health interventions provide an important means to curb the virus's reproduction number.

Conclusion:

I. Policy makers deem uncertainty as a major challenge, who must consider the consequences that relaxing any public health measures might lead to the reoccurrence of the disease.

3.3 Probabilistic Model for Quantitative Risk Assessment of COVID-19 [3]

Heitor O. Duarte, Paulo G. S. C. Siqueira, Alexandre C. A. Oliveira, Márcio J. C. Moura
Accepted: May 2020
Published on: Researchgate.net

In this paper, Quantitative Microbial Risk Assessment (QMRA) was carried out to predict the relative risks for future scenarios and the effect of containment measures from March 17th 2020 to March 16th 2021. The model was verified by comparing the results with real values of the six-week period from March 17 - April 28 2020.

Methodology: The proposed model is probabilistic in nature which allows for consideration of uncertainty. Quantitative Risk Assessment is mainly related to risk communication. Therefore, risk categories have been used in QMRA to make risk communication easier.

Results:

VIII. Based on the results, in the absence of interventions, Africa is likely to be the continent with the largest number of infected people in the future followed by Europe, South America, North America, Oceania and Asia.

Limitations:

I. Probability of a recovered individual being infected again is 0 in the proposed model Yet, there is evidence that recovered individuals may become infected again.

3.4 Dynamic model of COVID-19 disease with exploratory data analysis [4]

Michael O. Adeniyi, Matthew I. Ekum, Iluno C, Ogunsanya A. S, Akinyemi J. A, Segun I. Oke, Matadi M. B.

Accepted: 9th July 2020

Indexing: Science Direct

- The research conducted aims to find the effect of hygiene and educational campaign on curbing the spread of the pandemic.
- A non-linear mathematical model is developed to portray the effect of healthy sanitation and awareness on the transmission and dynamics of COVID-19.
- A basic reproduction threshold R_0 determines whether or not the disease will end eventually.

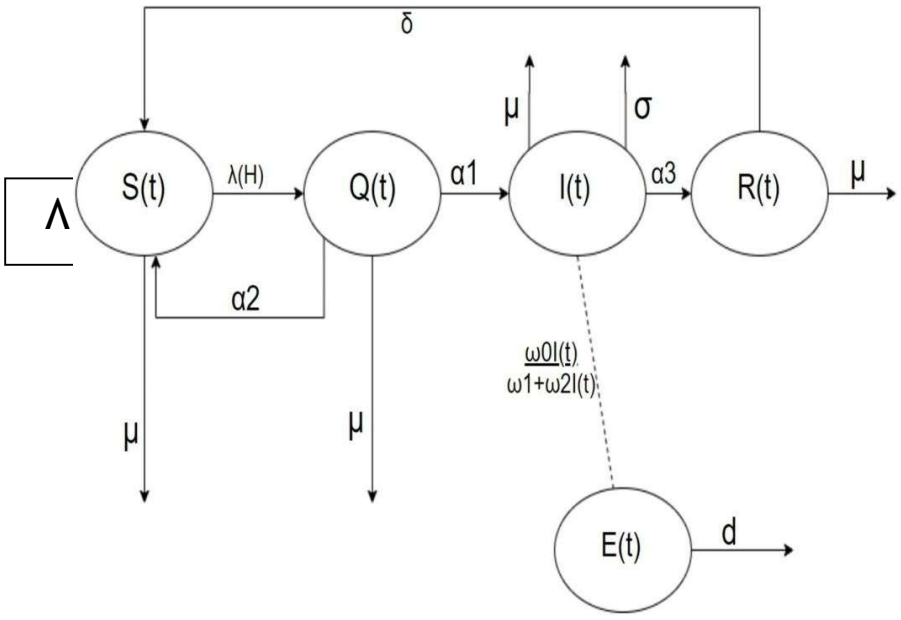
Model formulation:

SQIRES model is developed

Each variable is represented as a function of time because they vary with time.

S(t) – Susceptible, Q(t) – Quarantined, I(t) – Infectious, R(t) – Recovered humans, E(t) – Education.

Diagram of model between humans with the effects of education



Conclusion:

1. Restriction is the most sensitive parameters that increases basic reproduction number (R_0).
2. Testing and isolation centres should work full-fledged at airports and borders.
3. Probability of getting infected is high on coming in contact with infected person.
4. Transmission rate of the disease is high indicating the seriousness of the pandemic.
5. Promoting hygiene and education can reduce the spread.
6. Lack of human awareness leads to rapid spread of the virus.

3.5 Covid-19 future forecasting using supervised machine learning models [5]

Furqan Rustam, Aijaz Ahmad Reshi, (member, ieee),Arif mehmood, Saleem ullah, Byung-won on, Waqar Aslam, (member, ieee) and Gyu Sang Choi

Accepted: May 2020

Indexing: IEEE

Abstract:

- Linear Regression
- Least absolute shrinkage and selection operator
- Support Vector Machine
- Exponential Smoothing
- In this research paper the above 4 supervised machine learning models have been used as standard models and among those they have distinguished which model gives the best performance and also which model gives the average and poor performances according to the provided parameters.
- These above models were used to predict the upcoming patients affected with COVID-19.

Introduction:

- For a specific disease we can apply regression models and other models like neural networks to predict the future outcomes.
- How in this research paper the mentioned learning models are considered as the standard models and why particularly selected those models because those learning models have been tested with the datasets that are provided by Johns Hopkins.
- In this research paper they have clearly stated that, if the dataset size increased then the performance also increases.
- Learning models was evaluated based on the measurements such as R^2 – Score, $R2_{adjusted}$ Score MSE , RMSE , and MAE.

Conclusion:

- Exponential Smoothing is the best model for forecasting.
- LR (Linear Regression) and LASSO work better to forecast the future confirmed cases.
- Because of the ups and downs in the data set the SVM (Support Vector Machine) work poor to forecast the future predictions.
- In the days to come, the fatality will increase and recovery cases decreases.

3.6 A novel framework for COVID-19 case prediction through piecewise regression in India [6]

Apurbalal Senapati, Amitava Nag, Arunendu Mondal, Soumen Maji

Accepted: November 2020

Indexing: PubMed Central

Based on a type of linear regression curve, the author highlights the prediction of COVID-19 pandemic spread using piecewise linear regression assuming the number of infected cases

follows a linear pattern and then increases exponentially. The model has been fitted to the dataset to predicted the number of positive cases and recoveries for the states Maharashtra, Assam, Kerala, West Bengal and Delhi.

Methodology:

Piecewise linear regression has been used in the prediction model which is a special type of linear regression. As data does not follow a linear pattern and the entire data cannot be properly fitted with a single line, piecewise linear regression approach has been proposed which uses breakpoints (point where the trend changes) as the turning point which results in less error values.

Results:

Magnitude of relative error (MRE) and mean absolute percentage error (MAPE) are the metrics used to measure the accuracy of the regression model prediction. The MRE for the positive COVID-19 cases is quite low which signifies that the data are scattered close to the regression line. The MAPE value is small which indicates that the differences between actual and predicted values are small.

Challenges:

The main challenge is to find the number of breakpoints and the line segments to fit the data with less error values. Here, the breakpoints have been noted by observing point where the trend changes.

CHAPTER 4

DATA

Data is sourced from covid19india.org. Entire data was spread across twenty-six raw data files. The files were formed based on the dates on which the cases were reported.

4.1 Attributes in the dataset

There are twenty-one attributes in the first two raw data files which spread across the months of January, February, March and up until 26th April. These attributes are:

- Patient Number: This number is incremented sequentially for every new case discovered.
  State Patient Number: This is a state code given to patients. This attribute is left empty for a major portion of the dataset excluding a few records in the beginning.
- Date Announced: An important attribute which indicates the date on which the case was detected.
- Estimated Onset Date: This attribute is left empty throughout the data set. This could have been included for any further updates when the onset date of the disease could be found out.
- Age Bracket: Age of the patient is recorded. Over 90 percentage of the dataset does not hold data for this attribute.
- Gender: The gender of some of the patients are included in the dataset.
  Detected City: The city in which the patient resides.
- Detected District: District where the patient belongs to. This attribute plays an important role in gauging the spread of the disease across the country.
- Detected State: The state where the patient resides.
  State Code: Each state is given a unique code.
- Current Status: Indicates the current status of the patient. This attribute is the most important attribute in predicting future cases.
- Notes: Some of the patient details which indicate the source of infection is included.

- Contracted from which: Cases where there is a clear finding of transmission of disease from a preexisting patient is indicated.
- Nationality: The country to which the patient belongs to.
- Type of transmission: Indicates whether the patient was infected within the country or outside India.
- Status Change Date: The date when the patient changed from infected to the final state as indicated in the corresponding Current State attribute.
- Source_1, Source_2, Source_3: These three attributes show the source of the data indicated in each record.
- Num Cases: Number of cases included in each record. This attribute is 1 up until April 26th, 2020 but they are grouped after this date to include multiple cases of similar type on a given date.

` After 26th April,2020 there are only 20 attributes. Expected Onset Age attribute is dropped.

4.2 Attributes included for data pre processing

Every attribute in the dataset is not required for our problem for now. We have utilized six attributes:

- Date Announced
- Age Bracket
- Detected District
- Detected State
- Current Status
- Num Cases

These attributes are considered necessary to build our prediction models. The remaining attributes can be utilized for the data extraction functionality. The date announced attribute helps to predict day wise spread of the pandemic. Detected district and detected state are important to analyze the region wise spread of COVID-19. Current status is the most important attribute because it indicates whether the patient has recovered or deceased due to the disease. Num cases keeps count of patients of similar current status on a given date.

4.3 Data Pre-processing

The dataset is sourced from covid19india.org. This website uses state bulletins and official handles to update the data daily. We have used raw data files to analyze the demographics at a patient level. As of today, we have 26 raw data files till May 2nd 2021 with 6,12,401 rows and 22 columns.

The data present in some of the early raw data files are not completely clean and there are lot of inconsistency and also there are many irrelevant columns which are not required. Hence, we perform data pre-processing and appropriately create or choose the rows and columns which is required for our project and group them accordingly.

This is a snapshot of the first five rows of the dataset before data pre-processing.

```
First 5 rows of dataset df
```

Out[30]:

	Patient Number	State Patient Number	Date Announced	Estimated Onset Date	Age Bracket	Gender	Detected City	Detected District	Detected State	State code	...	Contracted from which Patient (Suspected)	Nationality	Type of transmission	Status Change Date
0	1.0	KL-TS-P1	30/01/2020	NaN	20	F	Thrissur	Thrissur	Kerala	KL	...	NaN	India	Imported	14/02/2020
1	2.0	KL-AL-P1	02/02/2020	NaN	NaN	NaN	Alappuzha	Alappuzha	Kerala	KL	...	NaN	India	Imported	14/02/2020
2	3.0	KL-KS-P1	03/02/2020	NaN	NaN	NaN	Kasaragod	Kasaragod	Kerala	KL	...	NaN	India	Imported	14/02/2020
3	4.0	DL-P1	02/03/2020	NaN	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	DL	...	NaN	India	Imported	15/03/2020
4	5.0	TS-P1	02/03/2020	NaN	24	M	Hyderabad	Hyderabad	Telangana	TG	...	NaN	India	Imported	02/03/2020

5 rows × 22 columns

As we can see from the above snapshot that there are many irrelevant columns and many NaN values which has to be taken care of. While performing data pre-processing, we divide our dataset into four quarters which are as follows:

i. Quarter 1: From 01/01/2020 to 31/03/2020 (January – March)
ii. Quarter 2: From 01/04/2020 to 30/06/2020 (April – June)
iii. Quarter 3: From 01/07/2020 to 30/09/2020 (July – September)
iv. Quarter 4: From 01/10/2020 to 31/12/2020 (October - December)

We then group the rows according to the date, calculate cumulatively recovered, confirmed, active and deceased cases for each date. We also add two more columns to include the number of states and districts where cases were found on a particular date. While creating a new data frame we encountered some problems which are as follows:

i. Handle missing district names: Sometimes we end up with lots of "unknown" districts as the bulletins do not contain district details. In such cases, we ignore the records with missing districts while counting the number of districts. There aren't many records with missing state name and district name, so it doesn't affect our dataset by a huge degree.

ii. Discrepancy in the early versions of the raw data files: Since there were lot of inconsistencies in the data reported by the state government specifically till 26th April 2020, we compare the cases with covid19india.org spread trends and hard enter the case values directly in the dataset till 26th April 2020. After 26th April, we have confirmed the cases to be consistent with the covid19india.org spread trends.

Function to find the confirmed, recovered, active and deceased cases along with the number of states and districts.

In [ ]:
```
# Load the values for quarter2, quarter3, quarter4
# Load second half of the dataframe
def load_sec_half(quarter, active):

    # Recovered cases for a particular day is found out after grouping by date
    recovered = quarter.loc[quarter["Current Status"] == "Recovered"]
    recovered = recovered.iloc[:,4]
    quarter["Recovered"] = recovered
    recovered = quarter.groupby(["Date Announced", "Recovered"]).sum()["Num Cases"].reset_index()

    # Confirmed cases for a particular day is found out after grouping by date
    confirmed = quarter.loc[(quarter["Current Status"] == "Hospitalized")]
    confirmed = confirmed.iloc[:,4]
    quarter["Confirmed"] = confirmed
    confirmed = quarter.groupby(["Date Announced", "Confirmed"]).sum()["Num Cases"].reset_index()

    # Deceased cases for a particular day is found out after grouping by date
    deceased = quarter.loc[quarter["Current Status"] == "Deceased"]
    deceased = deceased.iloc[:,4]
    quarter["Deceased"] = deceased
    deceased = quarter.groupby(["Date Announced", "Deceased"]).sum()["Num Cases"].reset_index()

    # Find total number of states and districts where cases were found on given date
    no_of_states = quarter.groupby(["Date Announced"])["Detected State"].nunique().reset_index()
    no_of_districts = quarter.groupby(["Date Announced"])["Detected District"].nunique().reset_index()

    #Creating a new dataframe with the necessary columns which are later updated
    updated_quarter = pd.DataFrame(columns = ['Date Announced', 'Confirmed', 'Active', 'Recovered', 'Deceased', 'No_of_states',
                                              'No_of_districts'])

    # Adding values into the new dataframe
    # Extracting values according to position
    updated_quarter["Date Announced"] = recovered.iloc[:,0]
    updated_quarter["Confirmed"] = confirmed.iloc[:,2]
    updated_quarter["Recovered"] = recovered.iloc[:,2]
    updated_quarter["Deceased"] = deceased.iloc[:,2]
    updated_quarter["No_of_states"] = no_of_states.iloc[:,1]
    updated_quarter["No_of_districts"] = no_of_districts.iloc[:,1]
    updated_quarter["Active"] = updated_quarter["Confirmed"] - updated_quarter["Recovered"] - updated_quarter["Deceased"]

    # Calculating cumulative of "Active" cases from previous records
    updated_quarter.at[0,'Active'] = updated_quarter.at[0,'Active'] + active
    updated_quarter['Active'] = updated_quarter['Active'].cumsum(axis = 0)

    return updated_quarter
```

Snapshot of five rows of the dataset after data pre-processing.

Out[33]:

	Date Announced	Confirmed	Recovered	Active	Deceased	No_of_states	No_of_districts
100	10-04-2020	871	151	6561	22	22	92
101	11-04-2020	854	186	7188	41	20	92
102	12-04-2020	758	114	7790	42	19	91
103	13-04-2020	1243	112	8894	27	21	104
104	14-04-2020	1031	167	9721	37	21	102

This is the first part of our data frame where we divide our raw dataset into four quarters. Next, we create a data frame based on the state and the district. We create 366 rows corresponding to 366 days of year 2020 for every district in a state and for all states. Then we calculate the confirmed, recovered and deceased cases for a particular district on a particular date.

Function to generate a data frame with 366 rows for all districts in a state for all states.

```
In [34]: # district_generator is the function to create and update the dataframes
         # for each district in the dictionary, dictionary_dataframe
         def district_generator(district, state, dictionary):

             # Create a dataframe with required attributes for the district
             district_dataframe = pd.DataFrame(columns = ['Date', 'State', 'District', 'Confirmed', 'Recovered', 'Deceased'])
             # adding dates for each day of the year
             district_dataframe['Date'] = pd.date_range(start='01/01/2020', end='31/12/2020')
             # update district as the input district
             district_dataframe['District'] = district
             # update state as the input state
             district_dataframe['State'] = state
             # extract only records with given state
             extracted_state = df.loc[df["State"] == state]
             # extract only records with given district
             extracted_district = extracted_state.loc[extracted_state["District"] == district]



             # change the datatype of the Date column to datetime to make comparisions efficient
             try:
                 extracted_district['Date'] = pd.to_datetime(extracted_district['Date'], format = '%Y-%m-%d')
             except:
                 extracted_district['Date'] = pd.to_datetime(extracted_district['Date'], format = '%Y/%m%d')

             extracted_district.reset_index()

             df1 = pd.concat([district_dataframe, extracted_district]).drop_duplicates(subset = ['Date'] , keep='last')

             df1 = df1.sort_values(by=["Date"])                                    # sorting by the dates

             # Every district will have dates where records are missing,
             # such cells are filled as DNA that is "Data Not Available"
             df1[["Confirmed", "Recovered", "Deceased"]] = df1[["Confirmed", "Recovered", "Deceased"]].fillna(0)
```

Cumulative confirmed, recovered and deceased cases in Kolar every day

```
In [31]: # Checking for the district of Kolar in Karnataka

         karnataka_districts['Kolar'][250:260]
```

Out[31]:

	Date	State	District	Confirmed	Recovered	Deceased
82821	2020-09-07	Karnataka	Kolar	4055	3281	66
83475	2020-09-08	Karnataka	Kolar	4085	3350	68
84129	2020-09-09	Karnataka	Kolar	4182	3403	69
84783	2020-09-10	Karnataka	Kolar	4286	3449	71
85437	2020-09-11	Karnataka	Kolar	4356	3510	71
86091	2020-09-12	Karnataka	Kolar	4409	3580	73
86745	2020-09-13	Karnataka	Kolar	4482	3632	75
87399	2020-09-14	Karnataka	Kolar	4539	3710	75
88053	2020-09-15	Karnataka	Kolar	4608	3795	77
88707	2020-09-16	Karnataka	Kolar	4709	3849	78

With this data frame, we will be able to understand the spread of the disease at state and district level which may be helpful during the development of probabilistic models.

The last data frame is based on the age bracket. The data frame which is created here will have details at the patient level where a bitmap of age (0-15, 16-25, 26-40, 41-60, Above 60) is created along with the current status of the patient in a particular state and district on a particular date.

Creating new columns (0-15, 16-25, 26-40, 41-60, Above 60)

```
In [11]: # assigning all the columns to 0 intially
bitmap_df['0-15'] = bitmap_df['15-25'] = bitmap_df['25-40'] = bitmap_df['40-60'] = bitmap_df['>60'] = 0

#assining '1' for the respective age brackets
bitmap_df.loc[bitmap_df['Age Bracket'] <= 15, '0-15'] = 1

bitmap_df.loc[(bitmap_df['Age Bracket'] > 15) & (bitmap_df['Age Bracket'] <= 25), '15-25'] = 1

bitmap_df.loc[(bitmap_df['Age Bracket'] > 25) & (bitmap_df['Age Bracket'] <= 40), '25-40'] = 1

bitmap_df.loc[(bitmap_df['Age Bracket'] > 40) & (bitmap_df['Age Bracket'] <= 60), '40-60'] = 1

bitmap_df.loc[bitmap_df['Age Bracket'] > 60, '>60'] = 1

bitmap_df = bitmap_df.drop(['Detected City','Num Cases'],axis = 1).reset_index()
```

Assigning 1 to the age group corresponding to the current status

```
In [73]: bitmap_df[85000:85010]
```

Out[73]:

	Date Announced	Age Bracket	Detected District	Detected State	Current Status	0-15	15-25	25-40	40-60	>60
85000	29/06/2020	55	Ballari	Karnataka	Hospitalized	0	0	0	1	0
85001	29/06/2020	48	Ballari	Karnataka	Hospitalized	0	0	0	1	0
85002	29/06/2020	28	Ballari	Karnataka	Hospitalized	0	0	1	0	0
85003	29/06/2020	4	Ballari	Karnataka	Hospitalized	1	0	0	0	0
85004	29/06/2020	47	Ballari	Karnataka	Hospitalized	0	0	0	1	0
85005	29/06/2020	34	Ballari	Karnataka	Hospitalized	0	0	1	0	0
85006	29/06/2020	27	Ballari	Karnataka	Hospitalized	0	0	1	0	0
85007	29/06/2020	26	Ballari	Karnataka	Hospitalized	0	0	1	0	0
85008	29/06/2020	52	Ballari	Karnataka	Hospitalized	0	0	0	1	0
85009	29/06/2020	42	Ballari	Karnataka	Hospitalized	0	0	0	1	0

This data frame helps in identifying the age groups which are the most and least affected and it may help during the development of probabilistic models.

4.4 Size of the Dataset

The dataset is huge and spans over 6,12,401 rows and 22 attributes. This size of the dataset enables our prediction model to predict cases to a good amount of accuracy.

CHAPTER 5

METHODOLOGY

The first phase of our capstone project was to obtain data which can be utilized to train machine learning models and consequently predict future cases. This required data preprocessing and handling missing values. In the second phase of our project, we trained the data we obtained from our previous phase on machine learning models. Four models were used to predict the confirmed cases:

- Piecewise Linear Regression
- Hierarchical Bayesian Model
- Exponential Smoothing
- XGBoost (Extreme Gradient Boosting)

5.1 Piecewise Linear Regression

Linear Regression also known as Segmented Regression is a method in regression analysis where the independent variable is partitioned into intervals and a separate linear line is fitted for individual segments. This type of regression is used when the data can be clustered into different groups, exhibit relationships in the region. The region between the segments is called break points.

A segmented analysis is where the data is of type (y, x), where y is the dependent variable and x is the independent variable. The results are in the form of equations,

y1 = A1*x + C1 for x < Break Point

y2 = A2*x + C2 for x > Break Point

where, y1 and y2 are the predicted values.

A1 and A2 are the regression coefficients.

C1 and C2 are the regression constants.

There can be n number of segments or break-points for a non-linear data. The data is clustered and modeled for different linear regression lines where the Least square Error is minimized among the segments.

5.1.1 Methodology:

Data of five districts in India were chosen for the prediction of cumulative daily confirmed cases. Districts are “Bengaluru Urban”, “Chennai”, “Delhi”, “Mumbai” and “Thiruvananthapuram”. Data of each district is trained for quarters 2, 3, and 4 of the year 2020 and first two months of the year 2021. Training data is cumulative in nature i.e. training dataset ‘t1’ is data of first two months of the year 2021, ‘t2’ is the combination of quarter4 of the year 2020 and first quarter1 of the year 2021, similarly ‘t4’ is combination of all four quarters of the data. Data of the first three days in the month of March 2021 is the testing data.

The n segment linear regression was modeled using the python package “pwlf” which finds the best fit for the data using Least Squared error as the metric. The last segment of the model is used to predict the future values of the data. Keeping Absolute error and the number of segments by which the over-fitting may occur, an optimal number of segments can be determined by brute-force approach where iteratively incrementing the line segments until (datapoints/2) and predicting the values on the test data. The iteration where the minimum Root Mean Squared Error (RMSE) is observed is the optimal number of line segments for the model. Due to computational feasibility the number of segments was manually chosen to be ‘3’ which was a good fit for most of the district’s data.
Cumulative confirmed cases were predicted using the fitted model of each district and tested against the actual confirmed cases. Confidence intervals for the predicted values were calculated with 95% confidence level.

5.1.2 Implementation:

Segregation of data based on districts and quarters.

(i) Training dataset of “Bengaluru Urban” for “t2”

```
dist_d['Bengaluru Urban']['t2']['df']
```

	Date	State	District	Confirmed
0	2020-10-01	Karnataka	Bengaluru Urban	237516
1	2020-10-02	Karnataka	Bengaluru Urban	241775
2	2020-10-03	Karnataka	Bengaluru Urban	245700
3	2020-10-04	Karnataka	Bengaluru Urban	250040
4	2020-10-05	Karnataka	Bengaluru Urban	252229
...	...	...	...	...
197	2021-04-16	Karnataka	Bengaluru Urban	522438
198	2021-04-17	Karnataka	Bengaluru Urban	533842
199	2021-04-18	Karnataka	Bengaluru Urban	546635
200	2021-04-19	Karnataka	Bengaluru Urban	556253
201	2021-04-20	Karnataka	Bengaluru Urban	570035

202 rows × 4 columns

Pseudo code for Fitting the models and predicting the values with the python package “pwlf”

```
#Piecewise Linear Regression
import pwlf

#Test data for Quarter 1 of 2021
q1_2021 = [x for x in range(737791, 737882, 1)]

x = np.array(yearly['Date Announced'])
y = np.array(yearly['Confirmed'])


#Initialize piecewise linear fit with x and y data
myPWLF = pwlf.PiecewiseLinFit(x,y)

#Fit the data for three line segments
res = myPWLF.fit(3)

#Predict for the determined points
xHat = np.linspace(min(x), max(x), num = len(x))
yHat = myPWLF.predict(xHat)

```

Model with the highest average accuracy for each district was chosen for the prediction of cumulative confirmed cases.

```
for dist in dist_d.keys():
    predicted_max = 0
    for train in dist_d[dist]:
        if (train != "best"):
            predicted = districts_model(dist_d[dist][train]['df'], 3,\
                                        quarter2_21.loc[quarter2_21['District'] == dist])
            if(predicted[:n]['Accuracy'].mean() > predicted_max):
                dist_d[dist]["best"] = predicted[:n].reset_index(drop = True)
                predicted_max = predicted[:n]['Accuracy'].mean()
```

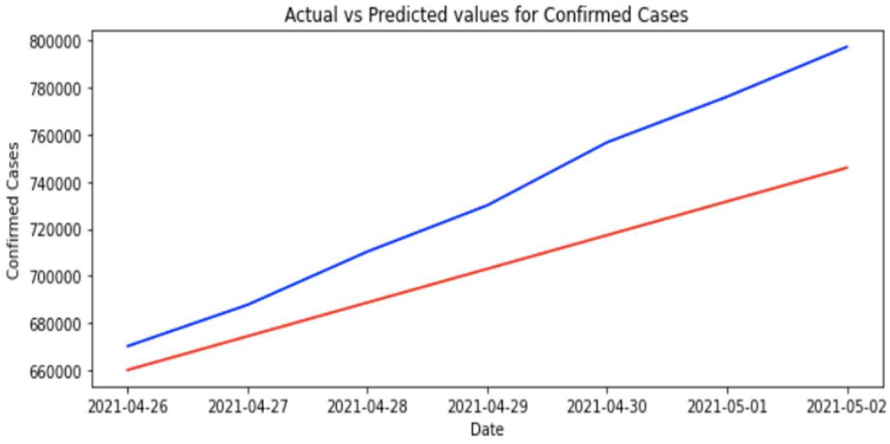
5.1.3 Results:

Best performing model was chosen for each district and a confidence interval is calculated with 95% confidence level. Results were visualized with the graph of actual vs predicted for each district of the testing dataset.

(i) Bengaluru Urban

```
dist_d['Bengaluru Urban']['best']
```

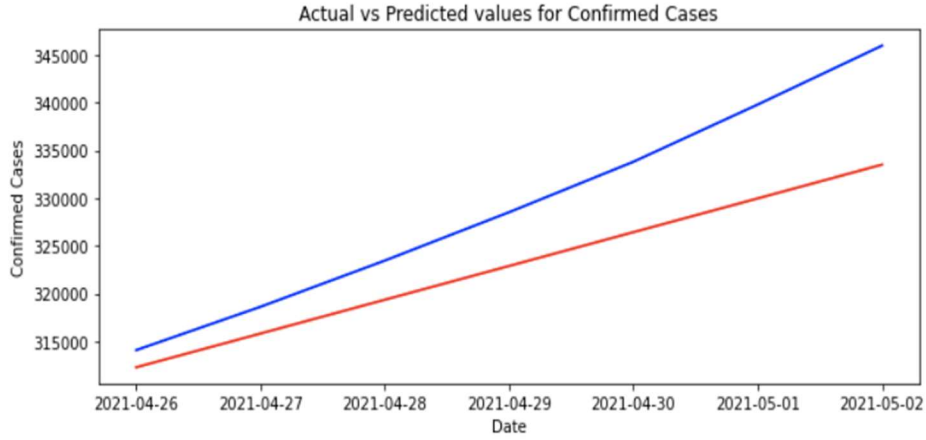
	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Min Predicted Confirmed	Max Predicted Confirmed	Accuracy	% of incorrect prediction
0	2021-04-26	670201	660092	10109	598272	721911	98.491647	1.508353
1	2021-04-27	687751	674400	13351	612580	736219	98.058745	1.941255
2	2021-04-28	710347	688709	21638	626889	750528	96.953883	3.046117
3	2021-04-29	729984	703017	26967	641197	764836	96.305809	3.694191
4	2021-04-30	756740	717326	39414	655506	779145	94.791606	5.208394
5	2021-05-01	776093	731634	44459	669814	793453	94.271434	5.728566
6	2021-05-02	797292	745943	51349	684123	807762	93.559574	6.440426



(ii) Chennai

```
dist_d['Chennai']['best']
```

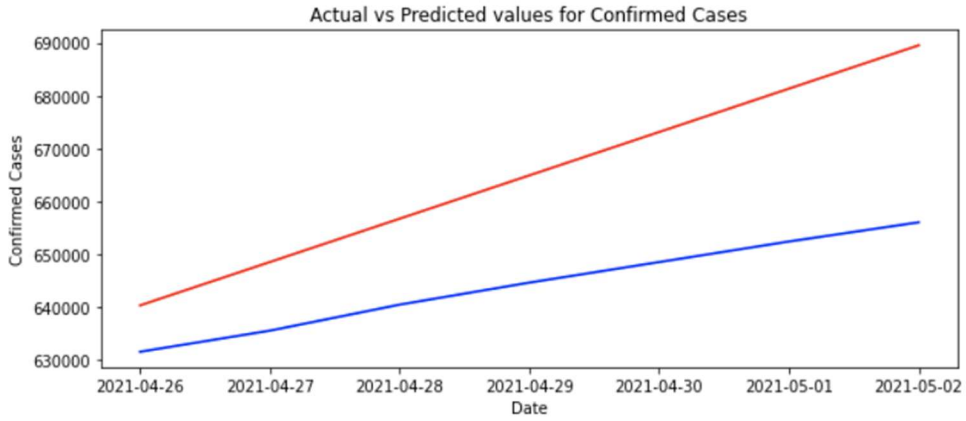
	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Min Predicted Confirmed	Max Predicted Confirmed	Accuracy	% of incorrect prediction
0	2021-04-26	314074	312275	1799	296986	327563	99.427205	0.572795
1	2021-04-27	318614	315813	2801	300524	331101	99.120880	0.879120
2	2021-04-28	323452	319352	4100	304063	334640	98.732424	1.267576
3	2021-04-29	328520	322890	5630	307601	338178	98.286254	1.713746
4	2021-04-30	333804	326429	7375	311140	341717	97.790620	2.209380
5	2021-05-01	339797	329967	9830	314678	345255	97.107096	2.892904
6	2021-05-02	345966	333506	12460	318217	348794	96.398490	3.601510



(iii) Delhi

```
dist_d['Delhi']['best']
```

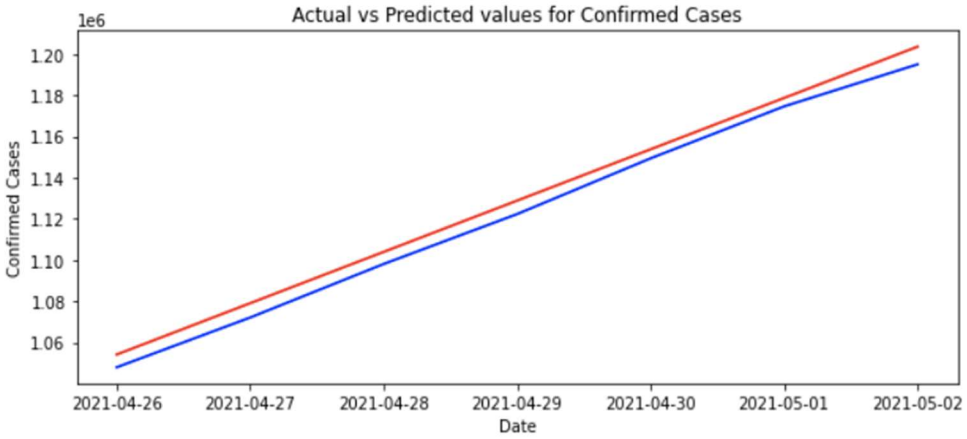
	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Min Predicted Confirmed	Max Predicted Confirmed	Accuracy	% of incorrect prediction
0	2021-04-26	1047916	1054112	6196	946517	1161706	99.408731	0.591269
1	2021-04-27	1072065	1079015	6950	971420	1186609	99.351718	0.648282
2	2021-04-28	1098051	1103919	5868	996324	1211513	99.465599	0.534401
3	2021-04-29	1122286	1128822	6536	1021227	1236416	99.417617	0.582383
4	2021-04-30	1149333	1153726	4393	1046131	1261320	99.617778	0.382222
5	2021-05-01	1174552	1178629	4077	1071034	1286223	99.652889	0.347111
6	2021-05-02	1194946	1203532	8586	1095937	1311126	99.281474	0.718526



(iv) Mumbai

```
dist_d['Mumbai']['best']
```

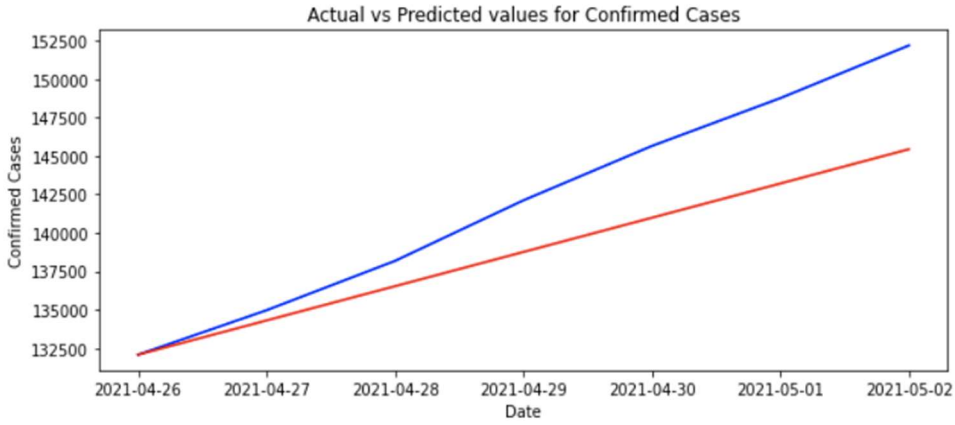
	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Min Predicted Confirmed	Max Predicted Confirmed	Accuracy	% of incorrect prediction
0	2021-04-26	631484	640267	8783	604795	675738	98.609149	1.390851
1	2021-04-27	635483	648477	12994	613005	683948	97.955256	2.044744
2	2021-04-28	640409	656687	16278	621215	692158	97.458187	2.541813
3	2021-04-29	644583	664897	20314	629425	700368	96.848505	3.151495
4	2021-04-30	648471	673107	24636	637635	708578	96.200910	3.799090
5	2021-05-01	652368	681317	28949	645845	716788	95.562474	4.437526
6	2021-05-02	655997	689527	33530	654055	724998	94.888696	5.111304



(v) Thiruvananthapuram

```
dist_d['Thiruvananthapuram']['best']
```

	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Min Predicted Confirmed	Max Predicted Confirmed	Accuracy	% of incorrect prediction
0	2021-04-26	132074	132076	2	122455	141696	99.998486	0.001514
1	2021-04-27	134966	134303	663	124682	143923	99.508765	0.491235
2	2021-04-28	138176	136530	1646	126909	146150	98.808766	1.191234
3	2021-04-29	142116	138757	3359	129136	148377	97.636438	2.363562
4	2021-04-30	145651	140983	4668	131362	150603	96.795079	3.204921
5	2021-05-01	148762	143210	5552	133589	152830	96.267864	3.732136
6	2021-05-02	152186	145437	6749	135816	155057	95.565295	4.434705



5.1.4 Discussions:

1. “Chennai” has the Least Absolute error among the all five districts followed by “Delhi”, “Mumbai”, “Bengaluru Urban”, “Thiruvananthapuram”
2. Due to the wake new of Coronavirus variant the cases have increased drastically and “Bengaluru Urban” is among the worst affected city.
3. Accuracy of all the districts is more than 95%.
4. Confidence interval is broad enough to fit in all the predicted values.

5.1.5 Conclusions:

1. Since this model has more than 95% accuracy it can be used to predict for more number of days when the trend of the data doesn’t change drastically.
2. If a greater number of segments are used then the model tends to overfit and predict values with less accuracy even for the smallest change in the trend of the data.
3. This model can be used for the data where the trend remains the same for a few weeks.
4. Optimizing the number of segments and confidence level can be considered as the future work of this model.

5.2 Hierarchical Bayesian Model

Hierarchical Bayesian model is a probabilistic model where parameters are nested with one another at different levels. The structure of the hierarchical model typically consists of parameters, priors and hyper priors. Hierarchical model is formed by combination of the sub-models and bayes theorem is used to instantiate all the sub-models with the observed data and will account for all the uncertainty that is present in the data by initializing probability distributions to the random variables. The result of this integration gives us a posterior distribution which is basically a probability distribution for a random variable and the parameters of the posterior distribution are estimated using the Bayesian method.

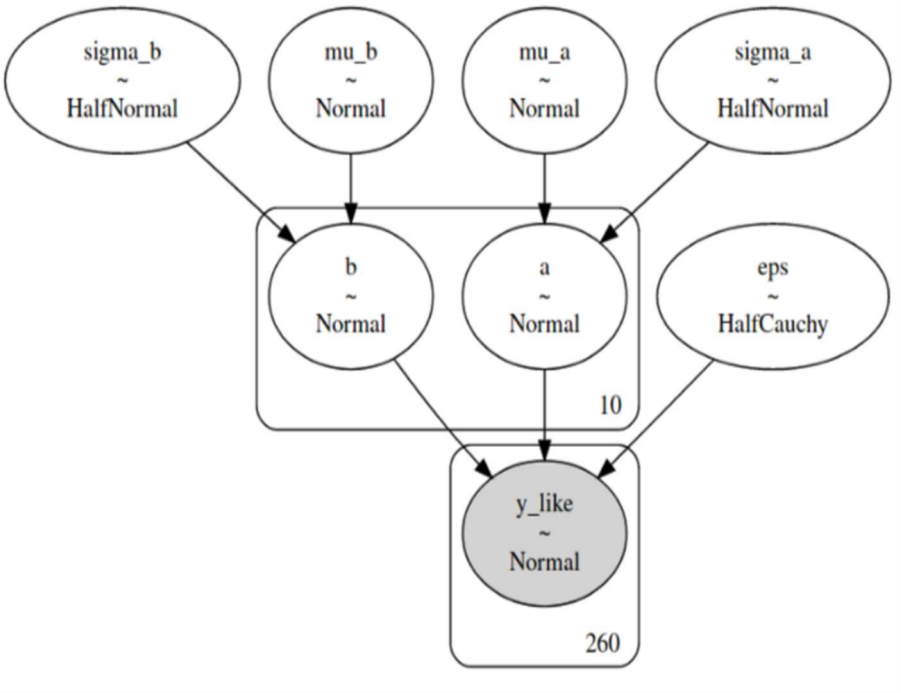
5.2.1 Components:

Two concepts are used in hierarchical Bayesian modelling to derive posterior distribution:
Hyperparameters - These are the parameters of the prior distribution.
Hyperpriors - These are the distributions of hyperparameters.

5.2.2 Method:

Data from five districts Mumbai, Bengaluru Urban, Chennai, Delhi and Thiruvananthapuram were chosen to predict the confirmed cases. Two training datasets were used which is from April 26th 2020 to Feb 28th 2021 in one training data and April 26th 2020 to April 25th 2021 in another training data and testing data is for a period of seven days from March 1st 2021 to March 7th 2021 and April 26th 2021 to May 2nd 2021 respectively. For each district, data points with 0 confirmed cases have been removed from the dataset

We have built a 3-stage hierarchical model by defining hyperpriors for the hyperparameters which in turn is used to construct prior distribution.
The graphical representation of the probabilistic model is show as follows:



Logistic regression function has been employed to model COVID-19 confirmed cases growth curve.

The model is represented as follows:

$$\mu = \frac{\alpha}{1 + exp(-\beta(t - t_0))}$$

where, Alpha is the maximum number of covid cases

Beta is the growth rate coronavirus spread

5.2.3 Implementation:

PyMC3 library was used to implement the model. It is a probabilistic programming framework used for creating user defined probabilistic models and allows for automatic Bayesian inference.

5.2.4 Model Definition:

We have used a logistic regression model with normal priors for the parameters and half-normal distribution as the prior for sigma.

Pseudo-code:

Logistic function to define the model

```
# Logistic regression function
def logistic(K, r, t, C_0):
    A = (K-C_0)/C_0
    return K / (1 + A * np.exp(-r * t))
```

5.2.5 Model Specification:

Here, r_mu, r_sigma, K_mu and K_sigma are hyperpriors to which normal and half-normal distributions have been assigned to and these introduce dependencies between each district's sub-models. r and K are the priors defined using the hyperpriors which will then be used to build the logistic regression model. Then, r, K, C_0 and t are passed to the logistic regression model with a pm.Lognormal distribution as the likelihood.

```python
# Model fitting using pymc3
bayesian_model = pm.Model("Bayesian Model")
with bayesian_model:
    BoundedNormal = pm.Bound(pm.Normal, lower=0.0)

    # Intercept
    C_0 = pm.Normal("C_0", mu=1000, sigma=100)

    # Growth rate
    r_mu = pm.Normal("r_mu", mu=0.4, sigma=0.1)
    r_sigma = pm.HalfNormal("r_sigma", 0.5)
    r = BoundedNormal("r", mu=r_mu, sigma=r_sigma, shape=n_districts)

    # Total number of cases
    K_mu = pm.Normal("K_mu", mu=800000, sigma=800000)
    K_sigma = pm.HalfNormal("K_sigma", 50000)
    K = pm.Gamma("K", mu=K_mu, sigma=K_sigma, shape=n_districts)

    # Likelihood error
    eps = pm.HalfNormal("eps")

    for i, district in enumerate(districts):

        df_district = sorted_data.loc[lambda x: (x.District == district)]

        # Loading train data
        t = pm.Data(district + "x_data", df_district['days_since_100_cases'])
        confirmed_cases = pm.Data(district + "y_data", df_district['Confirmed'])

        # Logistic regression
        growth = logistic(K[i], r[i], t, C_0)

        # Likelihood using Lognormal distribution
        pm.Lognormal(district, mu=np.log(growth), sigma=eps, observed=confirmed_cases)
```

5.2.6 Model Fitting:

After specifying the model, we obtain the posterior estimates for the unknown variables by using Markov chain Monte-carlo (MCMC) which include sampling algorithms known as No-U-Turn Sampler (NUTS). NUTS sampler is used for drawing samples from posterior distribution. PyMC3 automates the initialization of NUTS parameters such as samples, tunes etc. to reasonable values.

```
# NUTS sampler for drawing samples.
with hierarchical_model:
    trace = pm.sample()
```

```
Auto-assigning NUTS sampler...
Initializing NUTS using jitter+adapt_diag...
Multiprocess sampling (4 chains in 4 jobs)
NUTS: [Hierarchical Model_eps, Hierarchical Model_K, Hierarchical Model_K_sigma, Hierarchical Model_K_mu, Hierarchical Model_r,
Hierarchical Model_r_sigma, Hierarchical Model_r_mu, Hierarchical Model_C_0]
Sampling 4 chains, 1,513 divergences: 100%|████████████████████████████| 4000/4000 [4:37:34<00:00,  4.16s/draws]
There were 495 divergences after tuning. Increase `target_accept` or reparameterize.
There were 498 divergences after tuning. Increase `target_accept` or reparameterize.
There were 494 divergences after tuning. Increase `target_accept` or reparameterize.
There were 22 divergences after tuning. Increase `target_accept` or reparameterize.
The rhat statistic is larger than 1.4 for some parameters. The sampler did not converge.
The estimated number of effective samples is smaller than 200 for some parameters.
```

5.2.7 Predicting:

We use pm.sample_posterior_predictive to predict the confirmed cases for the given length of the district data.

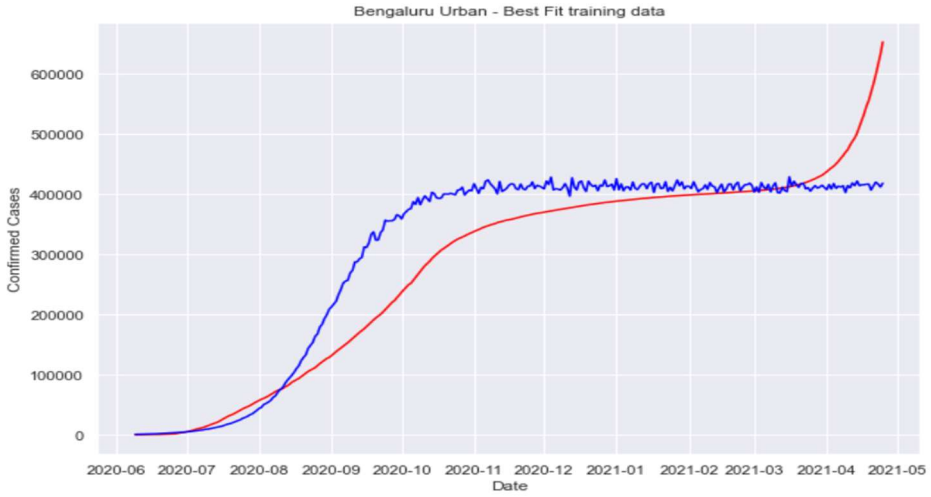
```
with bayesian_model:
    for district in districts:

        # Sample posterior predicting
        df_district = sorted_data.loc[lambda x: (x.District == district)]
        x_data = np.arange(0, len(df_district))
        y_data = np.array([np.nan] * len(x_data))
        pm.set_data({district + "x_data": x_data})
        pm.set_data({district + "y_data": y_data})

    ppc_hierarchical = pm.sample_posterior_predictive(trace)
```

```
100%|████████████████████████████████████████████████████████████| 1000/1000 [02:19<00:00,  7.15it/s]
```

5.2.8 Results:

Data frames with actual and predicted confirmed cases along with other metrics for five districts and graphs of actual vs predicted confirmed from March 1st 2021 to March 7th 2021 and April 26th 2021 to May 2nd 2021 have been plotted.

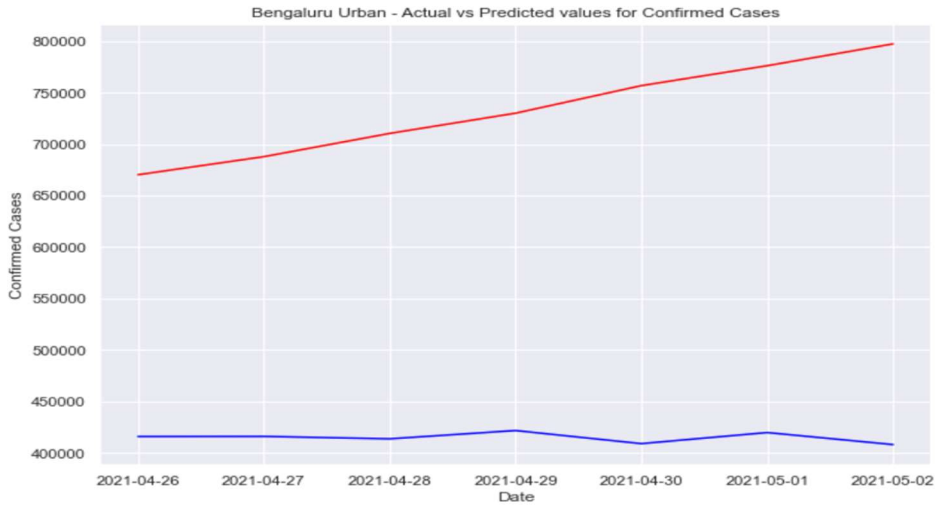
(i) Bengaluru Urban (Till February 28):



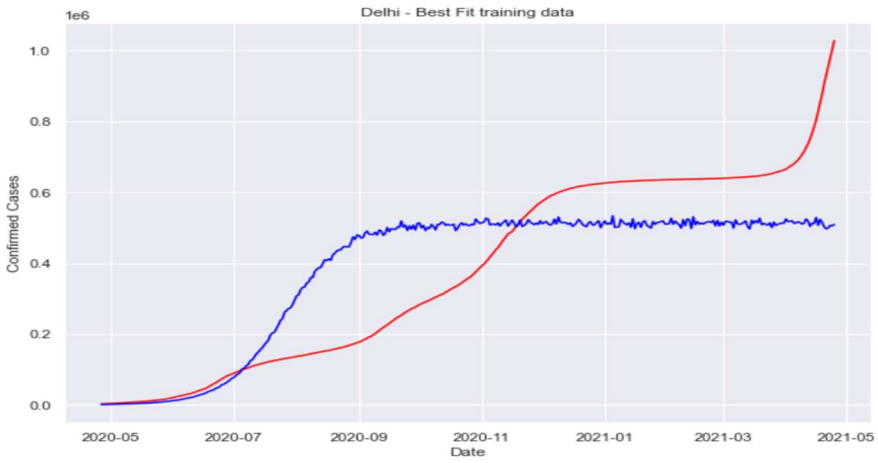
Actual confirmed, predicted confirmed, absolute error, accuracy, % of incorrect prediction for 7 days in Bengaluru Urban

Out[26]:

	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	670201	415909	254292	62.057353	37.942647
1	2021-04-27	687751	416016	271735	60.489334	39.510666
2	2021-04-28	710347	413673	296674	58.235341	41.764659
3	2021-04-29	729984	421709	308275	57.769622	42.230378
4	2021-04-30	756740	408998	347742	54.047361	45.952639
5	2021-05-01	776093	419762	356331	54.086559	45.913441
6	2021-05-02	797292	408088	389204	51.184259	48.815741



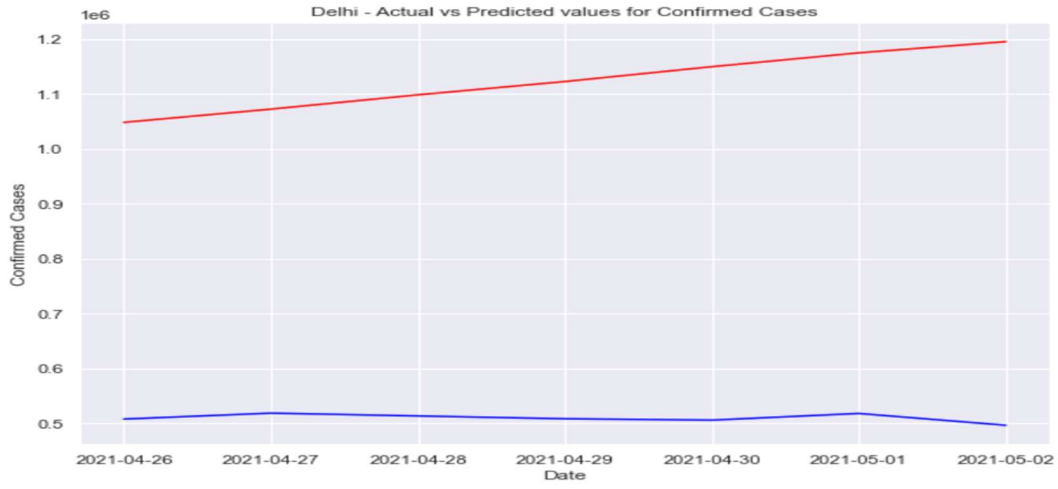
(ii) Delhi



Actual confirmed, predicted confirmed, absolute error, accuracy, % of incorrect prediction for 7 days in Delhi

Out[31]:

	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	1047916	507530	540386	48.432317	51.567683
1	2021-04-27	1072065	518282	553783	48.344270	51.655730
2	2021-04-28	1098051	513189	584862	46.736354	53.263646
3	2021-04-29	1122286	508049	614237	45.269120	54.730880
4	2021-04-30	1149333	505572	643761	43.988296	56.011704
5	2021-05-01	1174552	517715	656837	44.077657	55.922343
6	2021-05-02	1194946	496232	698714	41.527567	58.472433

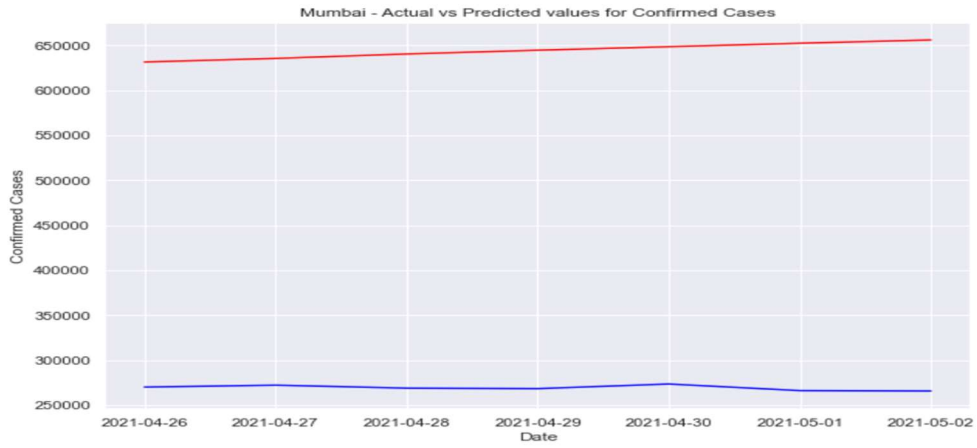


(iii) Mumbai

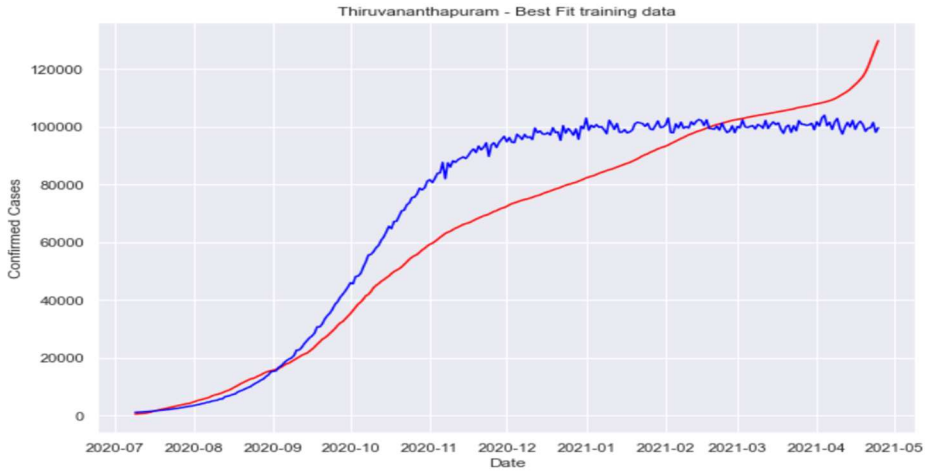
Actual confirmed, predicted confirmed, absolute error, accuracy, % of incorrect prediction for 7 days in Mumbai

Out[43]:

	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	631484	270045	361439	42.763554	57.236446
1	2021-04-27	635483	272186	363297	42.831358	57.168642
2	2021-04-28	640409	268879	371530	41.985512	58.014488
3	2021-04-29	644583	268358	376225	41.632808	58.367192
4	2021-04-30	648471	273459	375012	42.169812	57.830188
5	2021-05-01	652368	266172	386196	40.800898	59.199102
6	2021-05-02	655997	265722	390275	40.506588	59.493412



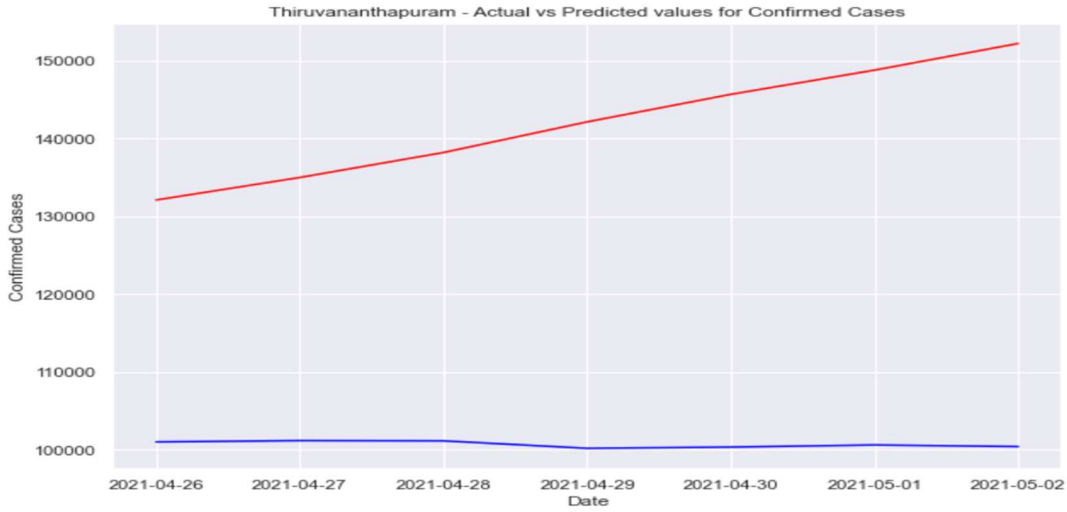
(iv) Thiruvananthapuram



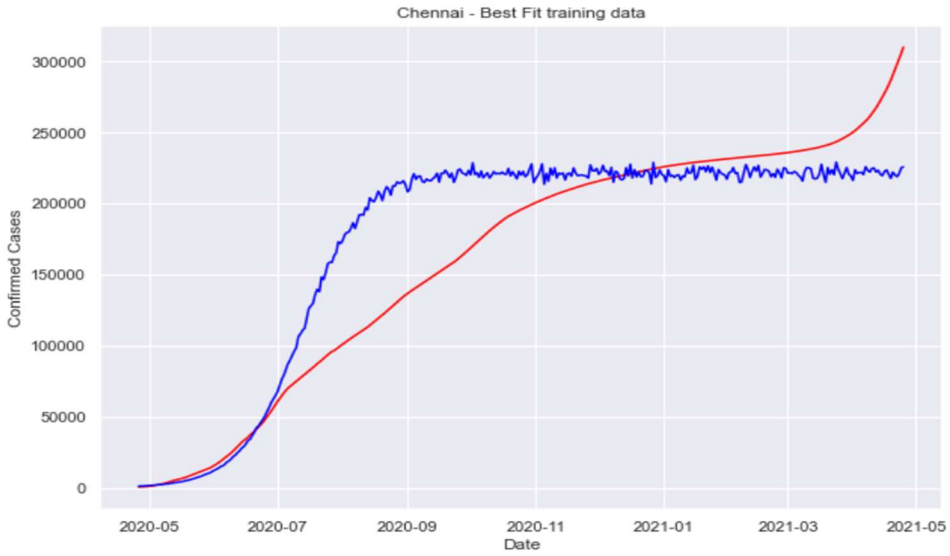
Actual confirmed, predicted confirmed, absolute error, accuracy, % of incorrect prediction for 7 days in Thiruvananthapuram

Out[35]:

	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	132074	100978	31096	76.455623	23.544377
1	2021-04-27	134966	101143	33823	74.939614	25.060386
2	2021-04-28	138176	101112	37064	73.176239	26.823761
3	2021-04-29	142116	100156	41960	70.474823	29.525177
4	2021-04-30	145651	100314	45337	68.872854	31.127146
5	2021-05-01	148762	100583	48179	67.613369	32.386631
6	2021-05-02	152186	100380	51806	65.958761	34.041239



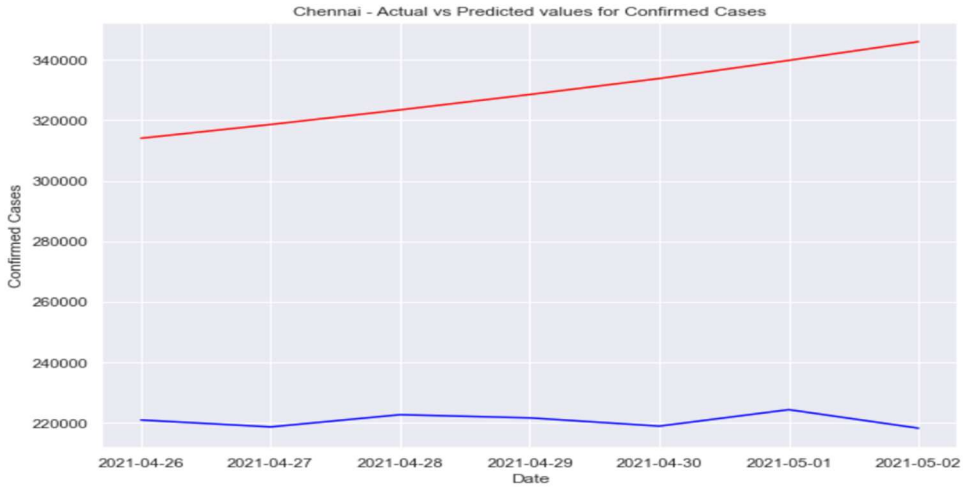
(v) Chennai



Actual confirmed, predicted confirmed, absolute error, accuracy, % of incorrect prediction for 7 days in Chennai

Out[39]:

	Date	Actual Confirmed	Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	314074	221027	93047	70.374179	29.625821
1	2021-04-27	318614	218747	99867	68.655803	31.344197
2	2021-04-28	323452	222777	100675	68.874825	31.125175
3	2021-04-29	328520	221729	106791	67.493303	32.506697
4	2021-04-30	333804	219007	114797	65.609459	34.390541
5	2021-05-01	339797	224440	115357	66.051201	33.948799
6	2021-05-02	345966	218324	127642	63.105623	36.894377



The model mainly follows a lognormal curve so the model wouldn't be able to predict the results accurately as we have seen in the case of Delhi and Mumbai due to the steep rise in the confirmed cases. As we can see from the graphs that Hierarchical Bayesian models tend to underestimate the number of confirmed cases which is true as the model fitted assumes that the cases follow a lognormal curve. We can resolve this by fitting different curves like the exponential curve but all the districts and states do not follow a set curve.

5.3 Exponential Smoothing

Exponential smoothing is a time series forecasting model where the weights are exponentially decreasing as the observation gets older i.e., recent observation is weighted more than the older observation for predicting the future values. Since the future confirmed cases depend on the current trend, Exponential smoothing was used to forecast the future confirmed cases.

$$F_t = \alpha * A_{t-1} + (1-\alpha) * F_{t-1} \quad \text{---------(1)}$$

F_t – Predicted Value

A_{t-1} – Previous Actual Value

F_{t-1} – Previous Predicted Value

α – Smoothing Constant (0, 1)

5.3.1 Methodology:

Data of five districts in India were chosen for the prediction of daily confirmed cases. Districts are “Bengaluru Urban”, “Chennai”, “Delhi”, “Mumbai” and “Thiruvananthapuram”. Data of each district is trained for quarters 2, 3, and 4 of the year 2020 and first two months of the year 2021. Training data is cumulative in nature i.e., training dataset ‘t1’ is data of first two months of the year 2021, ‘t2’ is the combination of quarter4 of the year 2020 and first quarter1 of the year 2021, similarly ‘t4’ is combination of

all four quarters of the data. Data of the first three days in the month of March 2021 is the testing data.

Model was fitted for each training dataset using a python module named "SimpleExponentialSmoothing" from stats models and the optimal Smoothing constant (α) for each individual model was calculated with least squared error as the metric. Models were tested against the testing dataset of each district and the model with least average accuracy was chosen for predicting COVID-19 daily confirmed cases in the respective districts.

Model was converted into a probabilistic model by calculating the confidence level with 95% confidence interval.

5.3.2 Implementation:

Segregation of data based on districts and quarters.

(i) Training dataset of "Bengaluru Urban" for "t1"

```
dist_d['Bengaluru Urban']['t1']['df']
```

	Date	State	District	Confirmed
0	2021-01-01	Karnataka	Bengaluru Urban	388850
1	2021-01-02	Karnataka	Bengaluru Urban	389193
2	2021-01-03	Karnataka	Bengaluru Urban	389657
3	2021-01-04	Karnataka	Bengaluru Urban	389955
4	2021-01-05	Karnataka	Bengaluru Urban	390348
...	...	...	...	...
105	2021-04-16	Karnataka	Bengaluru Urban	522438
106	2021-04-17	Karnataka	Bengaluru Urban	533842
107	2021-04-18	Karnataka	Bengaluru Urban	546635
108	2021-04-19	Karnataka	Bengaluru Urban	556253
109	2021-04-20	Karnataka	Bengaluru Urban	570035

110 rows × 4 columns

(ii) Training dataset of "Mumbai", for ('t2')

```
In [30]: dist_d['Mumbai']['t2']['df']
```

Out[30]:

	Date	State	District	Confirmed
0	2020-10-01	Maharashtra	Mumbai	2352
1	2020-10-02	Maharashtra	Mumbai	2440
2	2020-10-03	Maharashtra	Mumbai	2402
3	2020-10-04	Maharashtra	Mumbai	1190
4	2020-10-05	Maharashtra	Mumbai	1836
...	...	...	...	...
146	2021-02-24	Maharashtra	Mumbai	1167
147	2021-02-25	Maharashtra	Mumbai	1145
148	2021-02-26	Maharashtra	Mumbai	1035
149	2021-02-27	Maharashtra	Mumbai	987
150	2021-02-28	Maharashtra	Mumbai	1051

151 rows × 4 columns

Fitting the models and obtaining the optimal smoothing constant for each model.

```
In [31]: #Exponential
```

```
In [32]: from statsmodels.tsa.api import SimpleExpSmoothing
         %matplotlib inline
```

```
In [33]: #fit the models
         for district in dist_d:
             for train_data in dist_d[district]:
                 t = dist_d[district][train_data]
                 y_ses = pd.Series(t['df']['Confirmed'].to_list(),t['df']['Date'].to_list())
                 t['series_data'] = y_ses
                 t['fit'] = SimpleExpSmoothing(y_ses).fit()
                 t['alpha'] = t['fit'].model.params['smoothing_level']
             dist_d[district]["best"] = None
```

Pseudo code for training and testing the dataset.

```
In [55]: def train_val(df,alpha):
             pred = [0]
             for i in range(1,len(df)+1):
                 val = alpha*df[i-1] + ((1-alpha)*(pred[i-1]))
                 pred.append(int(val))
             return pred
```

```
In [56]: def predict_val(df_train,alpha,n):
             pred = df_train[len(df_train)-1]
             actual = alpha*pred
             for i in range(0,n-1):
                 val = alpha*actual + ((1-alpha)*(pred))
                 pred = val
                 actual = alpha*pred
                 df_train.append(int(val))
             return df_train
```

Model with the highest average accuracy for each district was chosen for the prediction of daily confirmed cases.

```
In [43]: for dist in dist_d.keys():
             predicted_max = 0
             for train in dist_d[dist]:
                 if (train != "best"):
                     predicted = predit(dist_d[dist][train]['df'],dist,dist_d[dist][train]['alpha'],n)
                     if(predicted[len(dist_d[dist][train]['df']):]['Accuracy'].mean() > predicted_max):
                         dist_d[dist]["best"] = predicted[len(dist_d[dist][train]['df']):]
                         predicted_max = predicted[len(dist_d[dist][train]['df']):]['Accuracy'].mean()
         #     dist_d[dist]['best'].to_csv(dist+".csv")
```

5.3.3 Results:

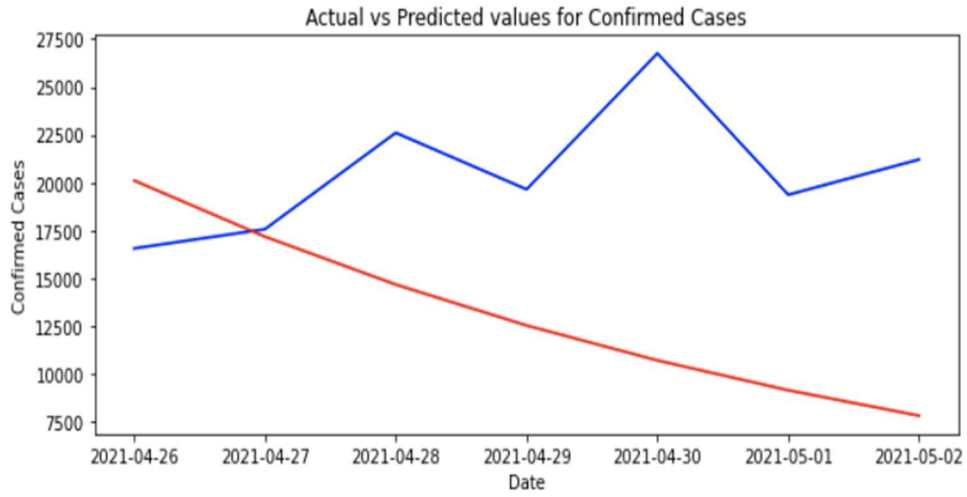
Best performing model was chosen for each district and a confidence interval is calculated with 95% confidence level. Results were visualized with the graph of actual vs predicted for each district of the testing dataset.

(i) Bengaluru Urban

```
dist_d["Bengaluru Urban"]["best"]
```

	Date	Actual Confirmed	Predicted Confirmed	Min Predicted Confirmed	Max Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	16545	20099	10955	29242	3554	78.519190	21.480810
1	2021-04-27	17550	17162	8018	26305	388	97.789174	2.210826
2	2021-04-28	22596	14654	5510	23797	7942	64.852186	35.147814
3	2021-04-29	19637	12513	3369	21656	7124	63.721546	36.278454
4	2021-04-30	26756	10685	1541	19828	16071	39.934968	60.065032
5	2021-05-01	19353	9124	-19	18267	10229	47.145145	52.854855
6	2021-05-02	21199	7791	-1352	16934	13408	36.751734	63.248266

```
plot_df_test(x2,y2,q1_2021,y_hat)
```

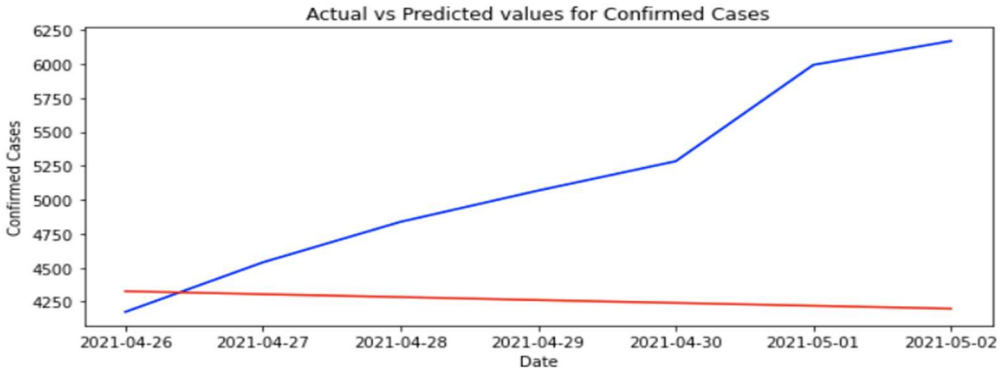


(ii) Chennai

```
dist_d["Chennai"]["best"]
```

	Date	Actual Confirmed	Predicted Confirmed	Min Predicted Confirmed	Max Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	4175	4327	1802	6851	152	96.359281	3.640719
1	2021-04-27	4540	4305	1780	6829	235	94.823789	5.176211
2	2021-04-28	4838	4284	1759	6808	554	88.548987	11.451013
3	2021-04-29	5068	4262	1737	6786	806	84.096290	15.903710
4	2021-04-30	5284	4241	1716	6765	1043	80.261166	19.738834
5	2021-05-01	5993	4220	1695	6744	1773	70.415485	29.584515
6	2021-05-02	6169	4199	1674	6723	1970	68.066137	31.933863

```
plot_df_test(x2,y2,q1_2021,y_hat)
```

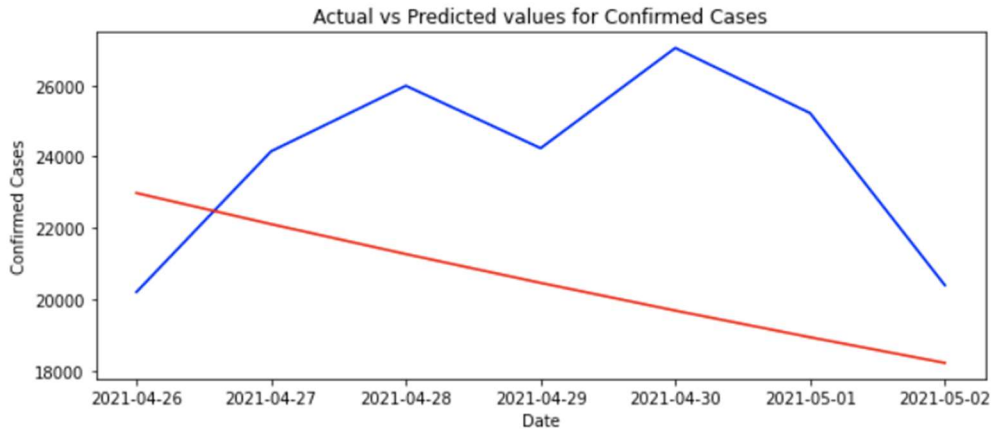


(iii) Delhi

```
dist_d["Delhi"]["best"]
```

	Date	Actual Confirmed	Predicted Confirmed	Min Predicted Confirmed	Max Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	20201	22979	7234	38723	2778	86.248206	13.751794
1	2021-04-27	24149	22106	6361	37850	2043	91.540022	8.459978
2	2021-04-28	25986	21267	5522	37011	4719	81.840222	18.159778
3	2021-04-29	24235	20459	4714	36203	3776	84.419228	15.580772
4	2021-04-30	27047	19682	3937	35426	7365	72.769623	27.230377
5	2021-05-01	25219	18935	3190	34679	6284	75.082279	24.917721
6	2021-05-02	20394	18216	2471	33960	2178	89.320388	10.679612

```
plot_df_test(x2,y2,q1_2021,y_hat)
```

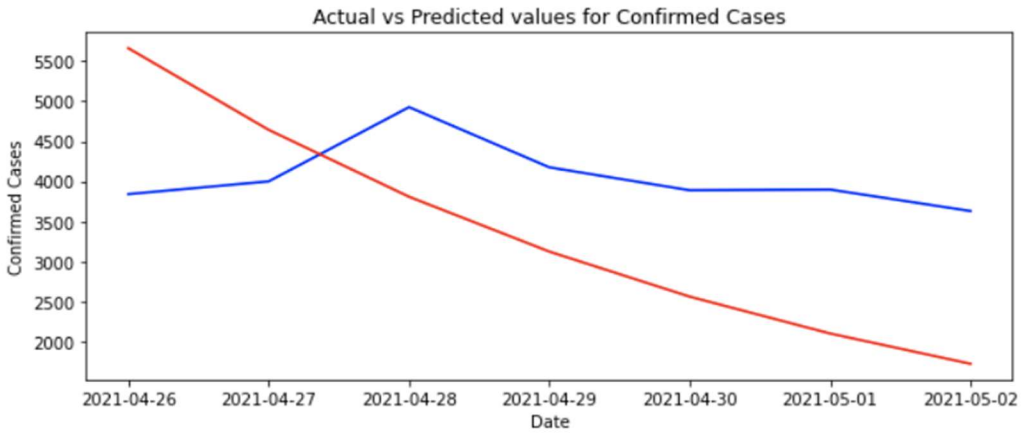


(iv) Mumbai

```
dist_d["Mumbai"]["best"]
```

	Date	Actual Confirmed	Predicted Confirmed	Min Predicted Confirmed	Max Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	3840	5661	-818	12140	1821	52.578125	47.421875
1	2021-04-27	3999	4642	-1837	11121	643	83.920980	16.079020
2	2021-04-28	4926	3807	-2672	10286	1119	77.283800	22.716200
3	2021-04-29	4174	3122	-3357	9601	1052	74.796358	25.203642
4	2021-04-30	3888	2560	-3919	9039	1328	65.843621	34.156379
5	2021-05-01	3897	2100	-4379	8579	1797	53.887606	46.112394
6	2021-05-02	3629	1722	-4757	8201	1907	47.451088	52.548912

```
plot_df_test(x2,y2,q1_2021,y_hat)
```

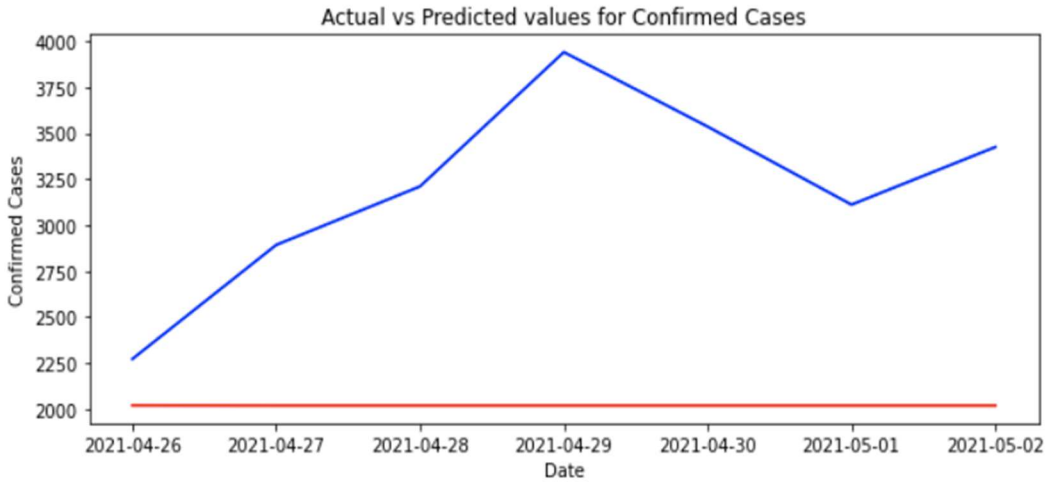


(v) Thiruvananthapuram

```
dist_d["Thiruvananthapuram"]["best"]
```

	Date	Actual Confirmed	Predicted Confirmed	Min Predicted Confirmed	Max Predicted Confirmed	Absolute Error	Accuracy	% of incorrect prediction
0	2021-04-26	2272	2020	933	3106	252	88.908451	11.091549
1	2021-04-27	2892	2019	932	3105	873	69.813278	30.186722
2	2021-04-28	3210	2019	932	3105	1191	62.897196	37.102804
3	2021-04-29	3940	2019	932	3105	1921	51.243655	48.756345
4	2021-04-30	3535	2019	932	3105	1516	57.114569	42.885431
5	2021-05-01	3111	2019	932	3105	1092	64.898746	35.101254
6	2021-05-02	3424	2019	932	3105	1405	58.966121	41.033879

```
plot_df_test(x2,y2,q1_2021,y_hat)
```



5.3.4 Discussions:

- The district “Mumbai” has the highest accuracy of 96% on day1 followed by “Chennai” with 95% accuracy.
- The district “Thiruvananthapuram” has recorded the least accuracy of 3% on day1. But has predicted cases with more than 95% in the following days.
- The minimum and maximum accuracy of “Bengaluru Urban” were 63% and 82% respectively
- Delhi has least average accuracy among the all five districts.

5.3.5 Conclusion:

- Few districts following the trend have higher accuracy than the ones which have fluctuating cases.
- Simple Exponential Model can be used only for predicting a few days because the trend of the model changes after a few days which affects the accuracy of the model.
- Confidence level of the models are high so we can work on optimizing the confidence level as our future work.

5.4 XGBoost (Extreme Gradient Boosting)

XGBoost is a machine learning model which can be used for regression and classification problems. It is a decision tree-based model which uses gradient boosting to improve its result with each iteration. XGBoost is widely used in the present world owing to its ease of use, high accuracy and good speed of prediction. This ensemble tree method boosts weak learners to provide higher accuracy with each iteration.

5.4.1 Training data

Data to train the model is obtained from districts.csv. The raw data is not suitable to train our model. Hence data has to be formatted. All districts are grouped together and every district containing missing data is removed. The data is trained from 26th April 2020 to 20th April 2021. XGBoost model cannot take datetime objects while training. The date column is hence converted to integer datatype.

Below is the snapshot of the cleaned and formatted train data set:

	Date	State	District	Confirmed	Recovered	Id
0	2020-04-26	Andaman and Nicobar Islands	Unknown	33	11	1
1	2020-04-27	Andaman and Nicobar Islands	Unknown	33	11	2
2	2020-04-28	Andaman and Nicobar Islands	Unknown	33	15	3
3	2020-04-29	Andaman and Nicobar Islands	Unknown	33	15	4
4	2020-04-30	Andaman and Nicobar Islands	Unknown	33	16	5
...	...	...	...	...	...	...
147455	2021-04-21	Punjab	Sri Muktsar Sahib	5911	4843	147456
147456	2021-04-22	Punjab	Sri Muktsar Sahib	6091	4906	147457
147457	2021-04-23	Punjab	Sri Muktsar Sahib	6295	4960	147458
147458	2021-04-24	Punjab	Sri Muktsar Sahib	6404	5029	147459
147459	2021-04-25	Punjab	Sri Muktsar Sahib	6704	5086	147460

147460 rows × 6 columns

5.4.2 Testing data

Test data starts from 21st April 2021. Test data contains date, state and district. An extra column - ForecastId is added so as to identify the predicted value belonging to which state and district. The date column is also modified to integer data type.

Below is the snapshot of cleaned and formatted test data set:

	Date	State	District	ForecastId
0	2021-04-26	Andaman and Nicobar Islands	Unknown	1
1	2021-04-27	Andaman and Nicobar Islands	Unknown	2
2	2021-04-28	Andaman and Nicobar Islands	Unknown	3
3	2021-04-29	Andaman and Nicobar Islands	Unknown	4
4	2021-04-30	Andaman and Nicobar Islands	Unknown	5
...	...	...	...	...
2823	2021-04-28	Uttarakhand	Udham Singh Nagar	2824
2824	2021-04-29	Uttarakhand	Udham Singh Nagar	2825
2825	2021-04-30	Uttarakhand	Udham Singh Nagar	2826
2826	2021-05-01	Uttarakhand	Udham Singh Nagar	2827
2827	2021-05-02	Uttarakhand	Udham Singh Nagar	2828

2828 rows × 4 columns

5.4.3 Model

Before the model is trained, we perform fit_transform on the training data so that we can scale the training data. The model then learns parameters such as mean and variance of the data in the training model.

'xgboost' is imported from XGBRegressor. For training the model we iterate through each district in each state in the training model, separate the confirmed cases and recovered cases into two different data frames and the remaining columns into a single data frame. Below is a snippet of the model being built:

```
from warnings import filterwarnings
filterwarnings('ignore')

# Predict data and Create submission file from test data
xout = pd.DataFrame({'ForecastId': [], 'Confirmed': [], 'Recovered': []})

for state in states:
    districts = X_xgtrain.loc[X_xgtrain.State == state, :].District.unique()
    #print(country, states)
    # check whether string is nan or not
    for district in districts:
        X_xgtrain_SD = X_xgtrain.loc[(X_xgtrain.State == state) & (X_xgtrain.District == district), ['Date', 'State', 'District',

        y1_xgtrain_SD = X_xgtrain_SD.loc[:, 'Confirmed']
        y2_xgtrain_SD = X_xgtrain_SD.loc[:, 'Recovered']

        X_xgtrain_SD = X_xgtrain_SD.loc[:, ['Date', 'State', 'District']]

        X_xgtrain_SD.State = le.fit_transform(X_xgtrain_SD.State)
        X_xgtrain_SD['District'] = le.fit_transform(X_xgtrain_SD['District'])

        X_xgtest_SD = X_xgtest.loc[(X_xgtest.State == state) & (X_xgtest.District == district), ['ForecastId', 'Date', 'State', '

        X_xgtest_SD_Id = X_xgtest_SD.loc[:, 'ForecastId']
        X_xgtest_SD = X_xgtest_SD.loc[:, ['Date', 'State', 'District']]

        X_xgtest_SD.State = le.fit_transform(X_xgtest_SD.State)
        X_xgtest_SD['District'] = le.fit_transform(X_xgtest_SD['District'])

        #models_C[country] = gridSearchCV(model, X_Train_CS, y1_Train_CS, param_grid, 10, 'neg_mean_squared_error')
        #models_F[country] = gridSearchCV(model, X_Train_CS, y2_Train_CS, param_grid, 10, 'neg_mean_squared_error')

        xmodel1 = XGBRegressor(n_estimators=1000)
        xmodel1.fit(X_xgtrain_SD, y1_xgtrain_SD)
        y1_xpred = xmodel1.predict(X_xgtest_SD)

        xmodel2 = XGBRegressor(n_estimators=1000)
        xmodel2.fit(X_xgtrain_SD, y2_xgtrain_SD)
        y2_xpred = xmodel2.predict(X_xgtest_SD)

        xdata = pd.DataFrame({'ForecastId': X_xgtest_SD_Id, 'Confirmed': y1_xpred, 'Recovered': y2_xpred})
        xout = pd.concat([xout, xdata], axis=0)
```

It is important to find the right parameter values for the XGBRegressor. Using gridSearchCV n_estimators equal to 1000 was found to give the best result and the remaining parameters such as eta (learning rate), max_depth (maximum depth of a tree) was found to be best at default.

‘xmodel1’ is the model used to predict the confirmed cases and ‘xmodel2’ is the model used to predict recovered cases. xout stores the result obtained after the prediction.

5.4.4 Result

xout presents the result obtained after the model predicts for the test data. It contains the predicted values, actual values, error of prediction and the percentage of error in prediction. Confirmed and recovered cases are both predicted. Below is a snippet of the xout data

frame:

	ForecastId	Confirmed	Recovered	Actual Confirmed	Actual Recovered	Error in Confirmed cases	Error in Recovered cases	% error Confirmed cases	% error Recovered cases
0	1	5664.998535	5466.998535	5716	5520	51.001465	53.001465	0.892258	0.960171
1	2	5664.998535	5466.998535	5764	5562	99.001465	95.001465	1.717583	1.708045
2	3	5664.998535	5466.998535	5816	5606	151.001465	139.001465	2.596311	2.479512
3	4	5664.998535	5466.998535	5875	5643	210.001465	176.001465	3.574493	3.118934
4	5	5664.998535	5466.998535	5949	5701	284.001465	234.001465	4.773936	4.104569
...	...	...	...	...	...	...	...	...	...
2823	2824	16342.997070	13271.997070	17872	13624	1529.002930	352.002930	8.555298	2.583697
2824	2825	16342.997070	13271.997070	18699	14315	2356.002930	1043.002930	12.599620	7.286084
2825	2826	16342.997070	13271.997070	19096	14315	2753.002930	1043.002930	14.416647	7.286084
2826	2827	16342.997070	13271.997070	19599	14605	3256.002930	1333.002930	16.613107	9.127031
2827	2828	16342.997070	13271.997070	20166	15033	3823.002930	1761.002930	18.957666	11.714248

2828 rows × 9 columns

5.4.5 Window Prediction

The results are presented in the form of 3 windows:

- 1-day window prediction (Date: 26/04/2021)
- 2-day window prediction (Date: 27/04/2021)
- 3-day window prediction (Date: 28/04/2021)

Five districts are selected to indicate the predicted values for these widows:

- Mumbai, Maharashtra
- Bengaluru Urban, Karnataka
- Chennai, Tamil Nadu
- Thiruvananthapuram, Kerala
- Delhi, Delhi

It is observed that the error in prediction increases as we move further into future cases. The first day provides the best prediction compared to other days. The percentage errors in

prediction are also compared.

Below is a snippet of the prediction for day 1:26/04/2021

	Date	State	District	Confirmed	Recovered	Actual Confirmed	Actual Recovered	Error in Confirmed cases	Error in Recovered cases	% error Confirmed cases	% error Recovered cases
0	2021-04-26	Karnataka	Bengaluru Urban	6.536558e+05	467312.906250	670201	471626	16545.187500	4313.093750	2.468690	0.914516
1	2021-04-26	Maharashtra	Mumbai	6.276438e+05	537943.812500	631484	544958	3840.187500	7014.187500	0.608121	1.287106
2	2021-04-26	Tamil Nadu	Chennai	3.098989e+05	273796.906250	314074	278330	4175.093750	4533.093750	1.329334	1.628676
3	2021-04-26	Delhi	Delhi	1.027715e+06	918874.812500	1047916	940930	20201.187500	22055.187500	1.927749	2.343978
4	2021-04-26	Kerala	Thiruvananthapuram	1.298020e+05	114031.976562	132074	114838	2272.023438	806.023438	1.720265	0.701879

Below is a snippet of the prediction for day 2:27/04/2021

	Date	State	District	ForecastId	Confirmed	Recovered	Actual Confirmed	Actual Recovered	Error in Confirmed cases	Error in Recovered cases	% error Confirmed cases	% error Recovered cases
0	2021-04-27	Karnataka	Bengaluru Urban	688	6.536558e+05	467312.906250	687751	475525	34095.187500	8212.093750	4.957490	1.726953
1	2021-04-27	Maharashtra	Mumbai	905	6.276438e+05	537943.812500	635483	552482	7839.187500	14538.187500	1.233579	2.631432
2	2021-04-27	Tamil Nadu	Chennai	1045	3.098989e+05	273796.906250	318614	282849	8715.093750	9052.093750	2.735314	3.200327
3	2021-04-27	Delhi	Delhi	1990	1.027715e+06	918874.812500	1072065	958792	44350.187500	39917.187500	4.136894	4.163279
4	2021-04-27	Kerala	Thiruvananthapuram	2326	1.298020e+05	114031.976562	134966	115850	5164.023438	1818.023438	3.826166	1.569291

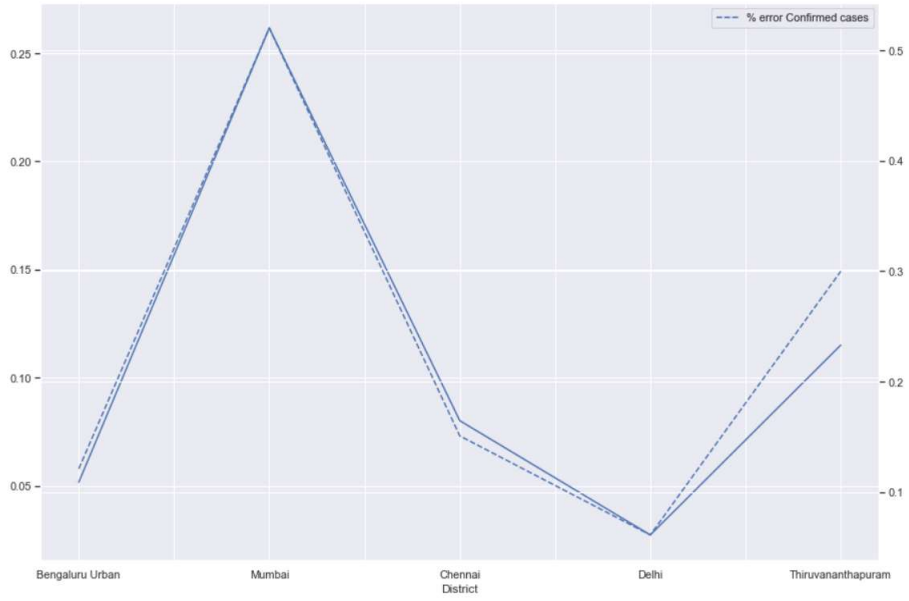
Below is a snippet of the prediction for day 3:28/04/2021

	Date	State	District	ForecastId	Confirmed	Recovered	Actual Confirmed	Actual Recovered	Error in Confirmed cases	Error in Recovered cases	% error Confirmed cases	% error Recovered cases
0	2021-04-28	Karnataka	Bengaluru Urban	689	6.536558e+05	467312.906250	710347	480055	56691.187500	12742.093750	7.980774	2.654299
1	2021-04-28	Maharashtra	Mumbai	906	6.276438e+05	537943.812500	640409	557948	12765.187500	20004.187500	1.993287	3.585314
2	2021-04-28	Tamil Nadu	Chennai	1046	3.098989e+05	273796.906250	323452	287496	13553.093750	13699.093750	4.190141	4.764968
3	2021-04-28	Delhi	Delhi	1991	1.027715e+06	918874.812500	1098051	982922	70336.187500	64047.187500	6.405548	6.515999
4	2021-04-28	Kerala	Thiruvananthapuram	2327	1.298020e+05	114031.976562	138176	117004	8374.023438	2972.023438	6.060404	2.540104

Comparison of percentage error in confirmed cases for day 1 vs day 2

--: date 2 prediction

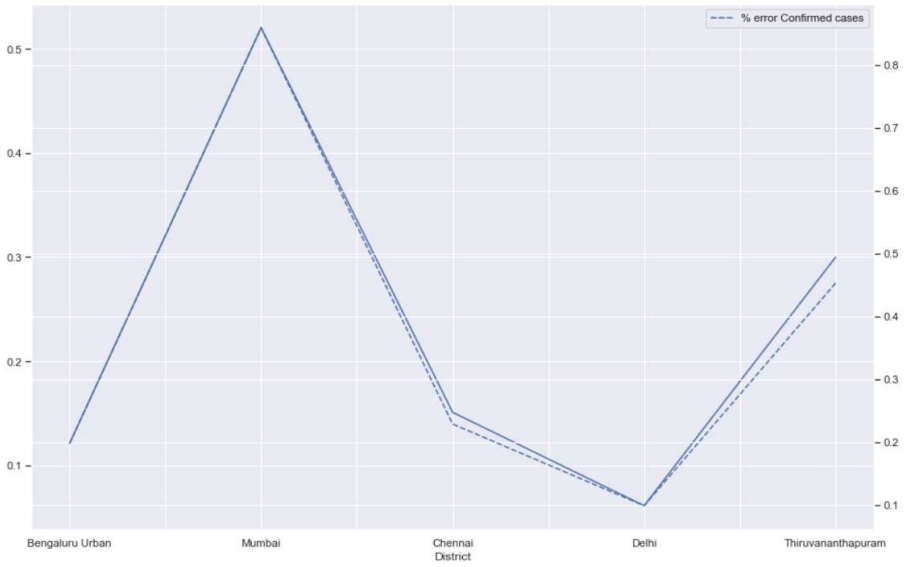
-: day 1 prediction



Comparison of percentage error in confirmed cases for day 2 vs day 3

--: date 3 prediction

-: day 2 prediction



5.4.6 Conclusion

- XGBoost provides excellent prediction for initial days of test data. For the first date of prediction the percentage error in prediction is less than 1% for all districts. However, the accuracy of prediction decreases as we move to future dates in the test data.
- After 30 days of prediction, the percentage of incorrect prediction is almost 30% indicating that the model is less accurate for distant future prediction.
- Every prediction using the model will need the best parameters possible to provide satisfactory results. These parameters such as eta or learning rate, n_estimators etc can be found out using gridSearchCV.

CHAPTER 6

RESULTS AND DISCUSSION

The literature survey showed that the probabilistic models couldn't consider the randomness such as hospital infrastructure, Antibodies developed in an individual, Common diseases among people. This has resulted in a quest for a probabilistic model which can include all the possible randomness mentioned above that helps in accurate prediction of No of cases of COVID-19.

Probabilistic Models couldn't add the randomness mainly because of lack of the data related to the randomness. The data was collected from a trusted open-source website, which had region wise daily cases recorded, state, district, and many others. The data was preprocessed and reformed to have 6 features. Errors in the data are corrected by editing the datasets manually. This helped in the model to predict more accurate results. The features added to the dataset are Confirmed Cases, Active Cases, Recovered Cases, Deceased, District count, State Count on each day for the period of four quarters. Features are created based on the input required for the prediction model and to meet the goal of the project i.e., to predict the number of cases. Data has more than 6 Lakh records which is considered to be very huge for any prediction model to provide accurate results. Data will be periodically updated for better prediction.

The general trend observed across the four models is that prediction accuracy reduces as we move towards future dates. All four models provide satisfactory results for the first 3 days. This has been showed using the 3-day window prediction for all four models. Piecewise Linear Regression provides the best result compared to other models. Hierarchical model provides a more realistic prediction but tend to underestimate the spread in certain cases as we can see the decrease in accuracy.

The results from the four models are shown in the website created. The webapp was created using plotly and dash components which includes HTML and CSS elements to visualize the results in the form of graphs. The model can be selected from the drop-down list. Following this the desired district is to be selected. A graph indicating the predicted confirmed cases and actual confirmed cases will be displayed. The data frame presented will contain error of prediction, percentage of incorrect prediction and accuracy of the model.

Snapshot of the webapp created:

Chapter 7

CONCLUSION AND FUTURE WORK

- **Piecewise Linear Regression**: This model takes the recent trend of the data into consideration and used it for prediction. Hence, this model provides very high accuracy even during second wave of COVID-19 pandemic because it was trained for sudden surge in COVID-19 cases. The only challenge we faced is the optimal number of line segments. With the metrics such as absolute error and computational time, we were able to arrive at four-line segments.
- **Hierarchical Bayesian Model**: This model has a high amount of accuracy when the curve follows a lognormal curve. The challenge we faced was to fit all the districts with a single lognormal curve. This is more observed when using training data from April 26th 2020 to April 25th 2021 since there is a huge surge of COVID-19.
- **Simple Exponential Smoothing**: This model performs best for prediction of 1 day. For the following days certain modifications were made in the algorithm to predict the future confirmed cases. This affected the accuracy of the model adversely. Hence, this model cannot be used for predicting COVID-19 cases for longer period.
- **XGBoost**: XGBoost is a good model for prediction because of its ease of use and accuracy. This model seemingly provides the best result because of the high accuracy for the initial days of test data with accuracies above 99%. The drawback of the model is that it does not follow the trend accurately. This leads to a deteriorating accuracy as we move forward to future days.

This study can be extended to future work by employing different methods to optimize the existing models. We can reduce the confidence interval window for better precision. Try out different curves to fit all the districts in Hierarchical Bayesian model. The parameters passed to the XGBoost model can be fine-tuned further and finding a way to implement optimal number of segments for piecewise linear regression. Furthermore, other types of data sources which has different parameters can used in our models.

REFERENCS

[1] Hemant Bherwani, Saima Anjum, Suman Kumar, Sneha Gautam, Ankit Gupta, Himanshu Kumbhare, Avneesh Anshul, Rakesh Kumar, "Understanding COVID-19 transmission through Bayesian probabilistic modeling and GIS-based Voronoi approach: a policy perspective," July 2020.

[2] Andrea L. Bertozzi, Elisa Franco, George Mohler, Martin B. Short, Daniel Sledge, "The challenges of modeling and forecasting the spread of COVID-19," July 2020.

[3] Heitor O. Duarte, Paulo G. S. C. Siqueira, Alexandre C. A. Oliveira, Márcio J. C. Moura, "Probabilistic Model for Quantitative Risk Assessment of COVID-19," May 2020.

[4] Michael O. Adeniyi, Matthew I. Ekum, Iluno C, Ogunsanya A. S, Akinyemi J. A, Segun I. Oke, Matadi M. B, "Dynamic model of COVID-19 disease with exploratory data analysis," July 2020.

[5] Furqan Rustam, Aijaz Ahmad Reshi, Arif mehmood, Saleem ullah, Byung-won on, Waqar Aslam, Gyu Sang Choi, "Covid-19 future forecasting using supervised machine learning models," May 2020.

[6] Apurbalal Senapati, Amitava Nag, Arunendu Mondal, Soumen Maji, "A novel framework for COVID-19 case prediction through piecewise regression in India," Nov 2020