

PROJECT FINAL REVIEW REPORT
on
**ANALYZING COVID-19 TRENDS USING
PYTHON**

*Submitted in partial fulfillment of the
requirement for the award of the degree of*

**Bachelors of Technology in
Computer Science Engineering**



(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

**Under The
Supervision of
Dr. S. Annamalai
Associate Professor**

Submitted By:

Ritvik Gautam (18021011613 / 18SCSE1010382)
Adarsh Kumar Singh (18021011279 / 18SCSE1010027)

**SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND
ENGINEERING GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
OCTOBER,
2021**



**SCHOOL OF COMPUTING SCIENCE AND
ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA**

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled “**ANALYZING COVID-19 TREND USING PYTHON**” in partial fulfillment of the requirements for the award of the **Bachelor of Technology** submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of 3 month, under the supervision of **Dr. S.Annamalai**, Associate Professor, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Ritvik Gautam(18SCSE1010382)

Adarsh Kumar Singh(18SCSE1010027)

**This is to certify that the above statement made by the candidates is correct
to the best of my knowledge.**

Dr. S.Annamalai

Associate Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voce examination of RITVIK GAUTAM(18SCSE1010382) & ADARSH KUMAR SINGH(18SCSE1010027) has been held on _____ and his/her work is recommended for the award of Bachelor of Technology.

Signature of Examiner(s)

Signature of Supervisor(s)

Signature of Project Coordinator


Signature of Dean

Date:

Place: Greater Noida

Communicated/Accepted/Published Proof

12/9/21, 9:48 PM IEEE IAS Global Conference on Emerging Technologies (GlobConET) : Submission (85) has been created. - ritvikgautam2412@g...

 Search mail

Compose

Inbox 5,120

Starred

Snoozed

Sent

Drafts 27


More

Meet


New meeting

Join a meeting

Hangouts

 RITVIK +

No recent chats
Start a new one



IEEE IAS Global Conference on Emerging

Microsoft CMT <email@msr-cmt.org>
to me

Hello,

The following submission has been created.

Track Name: Track VIII. AI, AIoT, IIoT, Deep Learning, and Macr

Paper ID: 85

Paper Title: Analysis of Covid-19 using Covid-19 Machine Learni

Abstract:
The Novel Corona virus, or COVID-19, epidemic in various parts will go down in history as one of the worst pandemics ever. The circumstances. Apart from global statistics, this study looks at Welfare, this study examines different trends and patterns noti June 24, 2020. The information could be investigated more in th

Created on: Thu, 09 Dec 2021 16:14:04 GMT

Last Modified: Thu, 09 Dec 2021 16:14:04 GMT

Authors:

- ritvikgautam2412@gmail.com (Primary)
- mohitkumar8133@gmail.com
- s.annamalai@galgotiasuniversity.edu.in

https://mail.google.com/mail/u/1/#inbox/FMfcgzGllMMVNmGLmtcgfcFsbCnVBMJJ 1/1

Table of Contents

✓ Table of Contents.....	i
✓ Abstract	ii
✓ List of Figures.....	iii
• Introduction	8
• Scope/objective	9
• Literature Reviews.....	10
• Project Diagrams.....	11-13
• Problem Formulation.....	14
• Tools used for implementation.....	15
• Feasibility Analysis.....	16
• Complete work plan layout.....	17
• Modules Description.....	18-24
• Merits of Proposed system.....	25
• Implementation and Testing.....	26-51
• Limitations.....	52
• Future Scope of the Project.....	53
• Results and Discussion.....	54-55
• Conclusions.....	56-57
• References.....	58

Abstract

EDA Data Analysis (EDA) is a data analysis field used to appear to represent information embedded in depth in a given data set. This method is widely used to generate clues from a given set of data. The current epidemic data set, COVID-19 is widely made available in a standard database. EDA can be used in this general database to generate consideration. In this paper, the data identification process is applied to the data that is available and is used to create patterns of better understanding of the epidemic effects in relation to the variables / labels assigned to the database.

A Web tool called Jupyter Notebook is used to make graphs using the Python language as it contains libraries used for the EDA process and the displays are displayed with symbols indicating high concentration. Based on the graphs obtained, we can draw conclusions about the current situation based on the available data, understand why certain variables increase / decrease in relation to each other and infer results from them.

List of Figures

Figure 1: Architecture Diagram of the System

Figure 2: UML Diagram

Figure 3: Use-case Diagram

Figure 4: Flow Diagram

Figure 5: Total Covid-19 cases month wise

Figure 6: Top 10 ages for the most number of infections.

Figure 7: State-wise total Covid-19 Cases in India.

Figure 8: Number of cases each day month-wise

Figure 9: Prediction using Linear Regression Algorithm

Figure 10: Prediction using Polynomial Algorithm

Figure 11: Prediction using Support Vector Regression Algorithm

Introduction

COVID-19 is a virus that belongs to the "Nidovirus family," or "Nidovirales," which includes the "Coronaviridae," "Artieviridae," and "Roiniviridae" families. It causes respiratory illness in humans, ranging from the common cold to more severe diseases like "Middle East Respiratory Syndrome(MERS)" and "Severe Acute Respiratory Syndrome(SARS)." Fever, fatigue, dry cough, aches and pains are the most typical symptoms or characteristics of COVID-19.

These signs or characteristics are present, although the person does not appear to be ill. People of any age group who have a medical history of high blood pressure, People with cardiovascular disease or diabetes are more likely to become infected, so anyone with a fever, cough, or breathing problems should seek medical help right away. COVID-19 is a “communicable” disease that spreads through droplets from the nose or mouth when an infected person coughs or exhales, which is why you should keep a distance of 1 metre (3 feet) from the sick person. COVID-19 is primarily distributed by contact rather than through the air, according to current research. Because so many people only had moderate symptoms, there's a good chance you'll catch COVID-19 from someone who only has a mild cough and doesn't seem ill.

COVID-19 protection and spread can be minimized by incorporating some simple and easy-to-follow precautions into daily habits, such as thoroughly cleaning hands with alcohol-based hand rub or washing them with soap and water, avoiding touching eyes, nose, and mouth as hands touch several surfaces that may be contaminated and hands could act as a carrier for COVID-19 and virus can enter our body, staying at home if you are sick, and most importantly, staying home if you are sick. Only pay attention to national and local authorities, as they will have the most up-to-date information on the issue.

When a student returned from Wuhan on January 30, 2020, India reported its first coronavirus case in Kerala, and the number of cases has been climbing dramatically since then. Using "Exploratory Data Analysis," this research examines the present trend of COVID-19 based on a set of criteria. Exploratory Data Analysis (EDA) is a method of examining data in order to extract relevant and actionable information. In any type of analysis, EDA is the revelatory phase.

Scope/objective

This study will be useful to the Government of India and various regions of India, Administrative Units of India, Indian Frontline health workers, researchers and scientists. This study will also be useful for foreign governance units to consider various factors related to COVID-19 outbreak in their regions.

Our main objective is to create a system that can provide a summary report, output (pie chart and bar graph), totals per location, growth rate, totals plot, world map. of cases in each country, SIR model (susceptible infected recovered).

This study attempts to work at a comprehensive level to analyse the spread of COVID19 in India.

To answer a variety of research questions, specific methods were used that included different data sets, data sources, modelling techniques and outcome variability.

Literature Survey

According to the various papers present in other's work, there are certain studies that keep an eye on trend insights and predicting for the Indian portion. Study on Indian Territory presents both short and long timeline trends.

These study reports use TS data from the JHU database and put before us forecasts with the help of the ARIMA model, exponential smoothing methods, the SER model, and the regression model. Furthermore, research in the India portion have previously focused on providing TSA based on aggregate data for the India portion.

Similar to others, there exist other mathematical models planned that were made to analyze COVID-19 outbreak trends in India. A model was presented to study the effect of social distinction on the basis of age and sex of sick people in Indian subcontinent. It studied the demographics of Indian, Italian, and Chinese countries and made recommendations for the weakest age groups and gender categories in each country.

Similarly, a study employed a linked structure to see if a certain node cluster was forming. Only the travelling information node was considered by the authors in order to determine what significant factors are influencing the return of Indian visitors to the Indian region. The study also presented us with SER models to examine the pace of virus dissemination in patients in the Indian region. Previous writers have suggested that we look into laboratories where testing takes place and infrared.

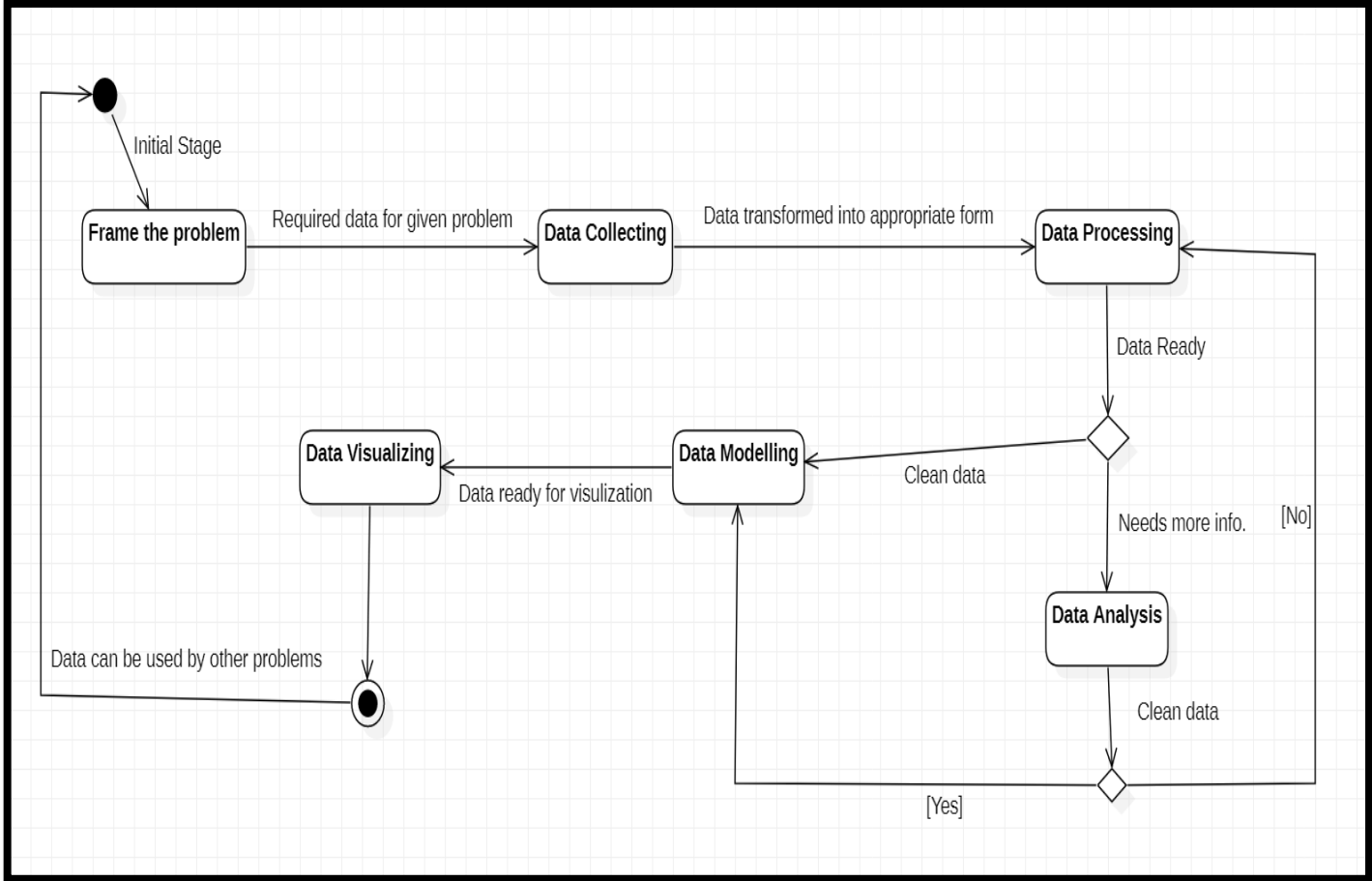
Some research has also brought to our attention the job of doctors and the health-care workforce. When compared to other nations such as Italy, Spain, and the United States, the function of health workers in India was less focused because the stages of virus expansion were in two or more phases.

Aside from India, there are models for other countries as well, primarily China, Italy, and America, due to the large number of afflicted people. The researchers used various mathematical forms to gauge the virus's spread, anticipate the number of infected persons, remark on each country's readiness to deal with COVID-19 propagation, and flatten curves under various scenarios.

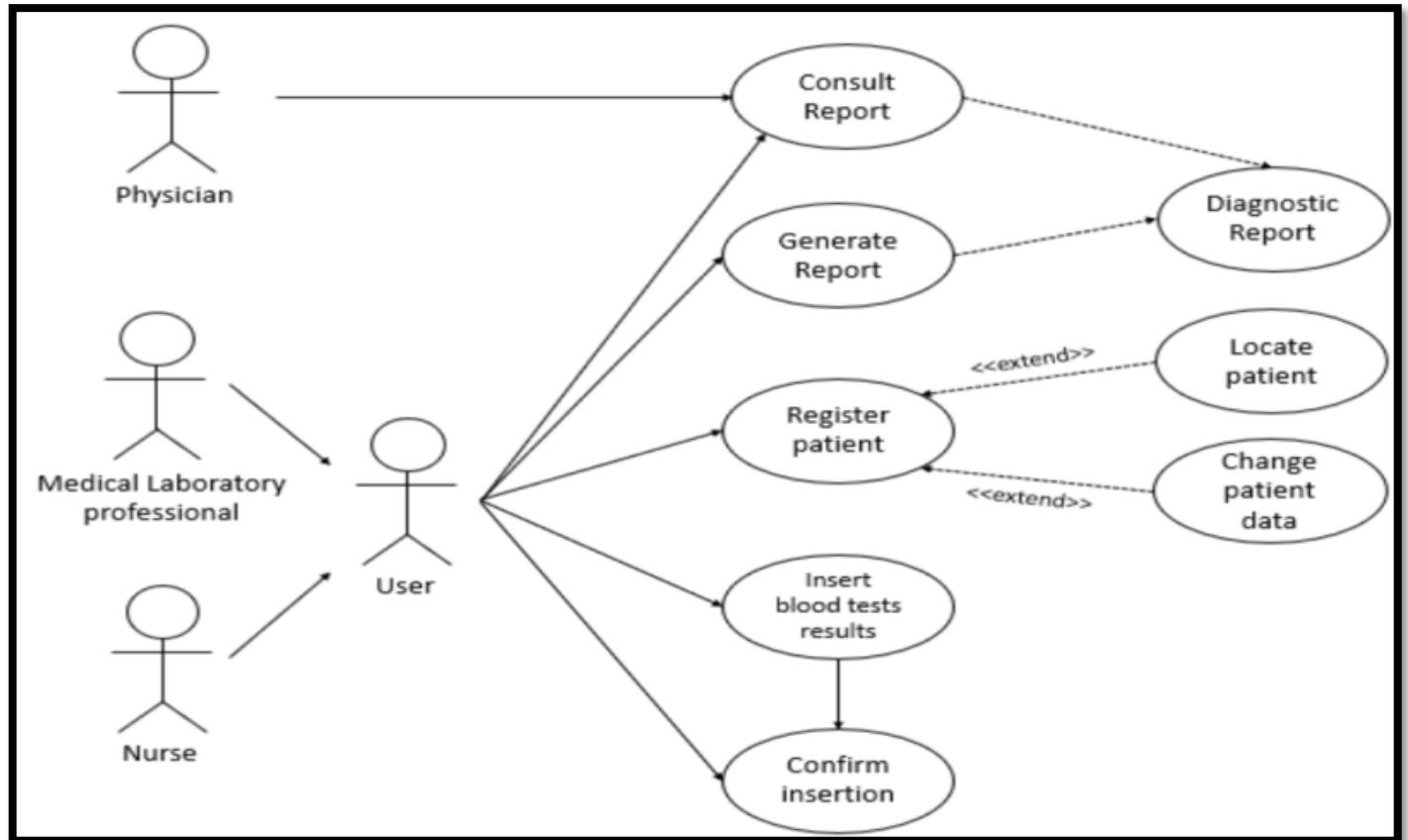
In terms of the research actions that took place in India, the study on the impact of various policy making efforts toward the management of this deadly virus is still being updated. This study tries to work extensively for the analysis of COVID spread in India.

Project Design

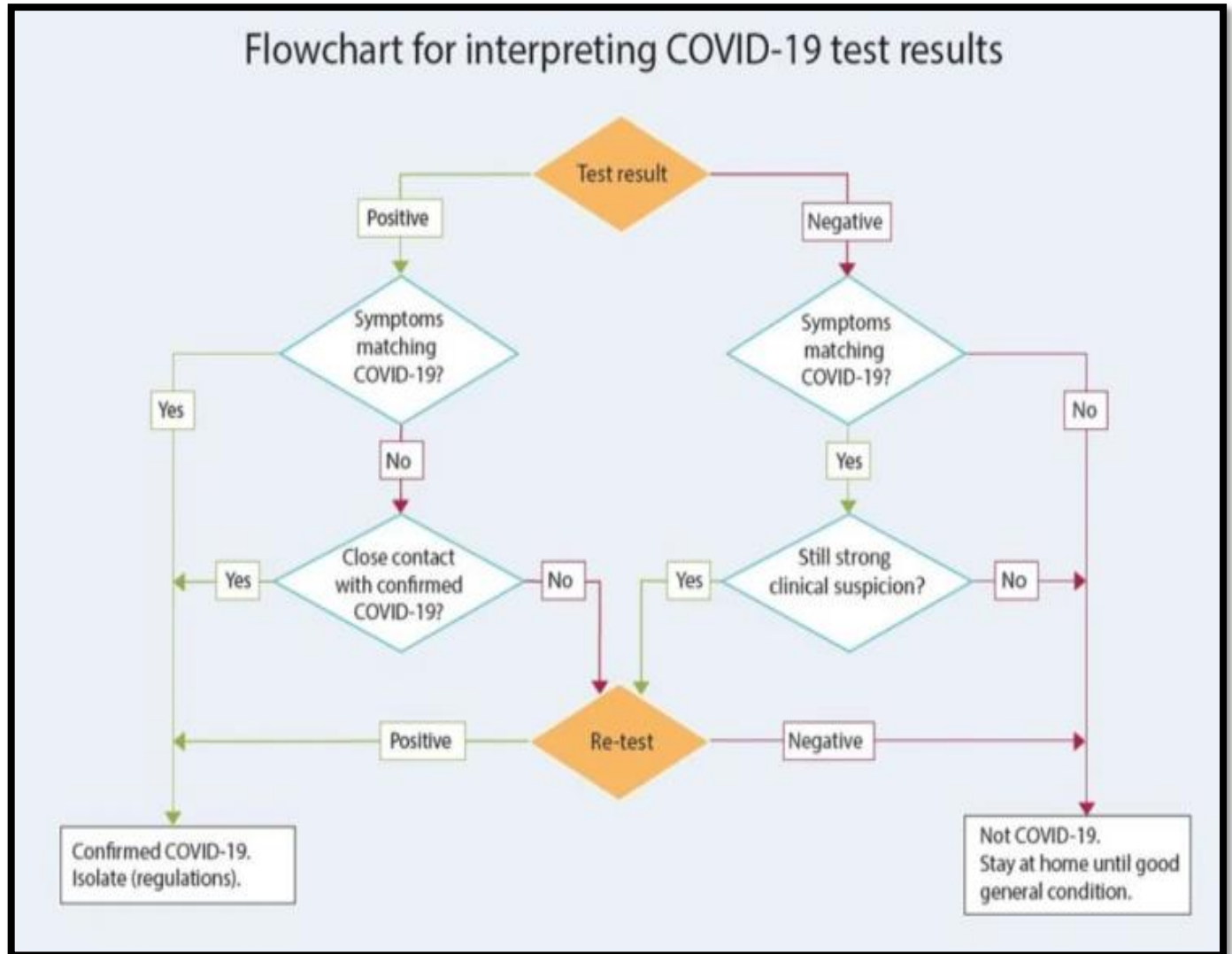
UML Diagram



Use-Case Diagram



Flow Diagram



Problem Formulation

The number of COVID-19 cases in India is increasing rapidly. National and local authorities have a difficult time compiling a pattern, analyzing and predicting the spread of COVID19 in India. The main purpose of this paper is to draw a statistical model to better understand the spread of COVID-19 in India by a careful study of the reported cases.

As we know some efforts have been made for the analysis of the spread of Covid-19. We aim to create a system that can provide a summary report, output (pie chart and bar graph), totals per location, growth rate, totals plot, incoming world map. of cases in each country, the SIR model (susceptible infected recovered) and predicts the appropriate number of cases in the future. Therefore, we can prepare and take steps to protect ourselves.

Tools used for implementation

Hardware Required:

- PC/LAPTOP.
- PC/LAPTOP having ram of at least 4 GB free space of 2 GB.
- PC/LAPTOP should have internet connectivity before executing the project.

Software Required:

- Python
- Jupyter Notebook (running on Anaconda)
- Library/packages – pandas, numpy, matplotlib, sklearn.

Feasibility Analysis

The corona virus epidemic (covid-19) presents a range of challenges for ongoing clinical trials, including new and non-standard causes of disappearance, including essentially high rates of missing outcome data. The International Drug Trial Guidelines advise testers to review plans for dealing with missing data in conduct and statistical analysis, but lack clear recommendations. Since the virus is new to us and little is known about it, there may be discrepancies in predictions because the data is not very accurate and therefore not giving an approximate number. Minor to major changes may occur in infected people in the future.

Complete Work Plan Layout

• 1st Phase

We will start by writing the programming part on Jupyter Notebook on covid-19 data analysis in which first we will install some libraries(Pandas, matplotlib.pyplot, matplotlib, plotly etc) from packages which will help in understanding the analysis of the covid-19. In this analysis first we will be storing inbuilt data functions of particular libraries in function by saving it and it will give information about the confirmed cases, cured cases, total per location, totals plot, deaths of people through covid-19 pandemic along with names of cities with their latitudes and longitudes giving us the exact location. Then we will be producing a timeseries summary report of top 10 countries which has the following details: For Confirmed Cases and For Death Cases we will represent:

- Countries and Cities reported
- Graphical outputs showing in the form of graphs and pie-chart
- Comparison of cases with global percentage and for predicting upcoming threat.

• 2nd Phase

Gathering multiple review papers i.e. about 50 and then we will be skimming through them, picking important points and connecting the gaps between them and in the final phase we will use time series forecasting in order to predict the future of Covid - 19 in India.

Modules Description

Data Collection

Data collection is defined as the procedure of collecting, measuring and analyzing accurate insights for research using standard validated techniques. A researcher can evaluate their hypothesis on the basis of collected data. In most cases, data collection is the primary and most important step for research, irrespective of the field of research. The approach of data collection is different for different fields of study, depending on the required information.

The most critical objective of data collection is ensuring that information-rich and reliable data is collected for statistical analysis so that data-driven decisions can be made for research.

Data Analysis

Data analysis is the practice of working with data to glean useful information, which can then be used to make informed decisions.

As the data companies have available to them continues to grow in both amount and complexity, so does the need for an effective and efficient process by which to harness the value of that data. The analysis method typically moves through several iterative phases. Let's take a closer look at each.

Identify the business question you'd like to answer. What problem is the company trying to solve? What do you need to measure, and how will you measure it?

Collect the raw data sets you'll need to help you answer the identified question. Data

collection might come from internal sources, like a company's client relationship management (CRM) software, or from secondary sources, like government records or social media application programming interfaces (APIs).

Clean the data to prepare it for analysis. This often involves purging duplicate and anomalous data, reconciling inconsistencies, standardizing data structure and format, and dealing with white spaces and other syntax errors.

Analyze the data. By manipulating the data using various data analysis techniques and tools, you can begin to find trends, correlations, outliers, and variations that begin to tell a story. During this stage, you might use data mining to discover patterns within databases or data visualization software to help transform data into an easy-to-understand graphical format.

Interpret the results of your analysis to see how well the data answered your original question. What recommendations can you make based on the data? What are the limitations to your conclusions?

Prediction using ML

“Prediction” refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, such as whether or not a customer will churn in 30 days. The algorithm will generate probable values for an unknown variable for each record in the new data, allowing the model builder to identify what that value will most likely be.

The word “prediction” can be misleading. In some cases, it really does mean that you are predicting a future outcome, such as when you’re using machine learning to determine the next best action in a marketing campaign. Other times, though, the “prediction” has to do with, for example, whether or not a transaction that already occurred was fraudulent. In that case, the transaction already happened, but you’re making an educated guess about whether or not it was legitimate, allowing you to take the appropriate action.

Algorithms used in this research in order to study and analyze the Covid-19 trends in order to predict the upcoming situations are:

- 1) Linear Regression Algorithm
- 2) Polynomial Regression Algorithm
- 3) Support Vector Regression Algorithm

i) Linear Regression

Because of its straightforward representation, linear regression is a popular model. The representation is a linear equation that combines a collection of input values (x), with the solution being the projected output for that set of input values (y). As a result, both the input (x) and output (y) values are numeric.

Each input value or column is assigned one scale factor, referred to as a coefficient and denoted by the capital Greek letter Beta in the linear equation (B). One more coefficient is added, which gives the line an extra degree of freedom (for example, going up and down on a two-dimensional plot) and is known as the intercept or bias coefficient.

For example, in a simple regression problem (with only one x and one y), the model would have the following form:

$$y = B_0 + B_1 * x$$

When there are several inputs (x) in higher dimensions, the line is called a plane or a hyper-plane. As a result, the representation is the equation's form as well as the coefficients' specific values (e.g. B0 and B1 in the above example).

The complexity of a regression model, such as linear regression, is frequently discussed. The number of coefficients utilized in the model is referred to as this.

When a coefficient becomes zero, it effectively removes the input variable's influence on the model and, as a result, the model's prediction (0 * x = 0). This is important to consider when considering regularization strategies, which alter the learning algorithm to minimize the complexity of regression models by exerting pressure on the absolute magnitude of the coefficients and driving some to zero. By applying Linear Regression algorithm on our accurate dataset we achieve a accuracy rate of 82 percent.

ii) **Polynomial Regression**

Polynomial Regression is a regression approach that uses an nth degree polynomial to represent the connection between a dependent(y) and independent variable(x). The equation for polynomial regression is as follow:

$$y = b_0 + b_1x_1 + b_2x_1^2 + b_3x_1^3 + \dots + b_nx_1^n$$

In machine learning, it's also known as the specific case of Multiple Linear Regression. Because we turn the Multiple Linear Regression equation into Polynomial Regression by adding certain polynomial terms.

It's a linear model that's been tweaked a little to improve accuracy.

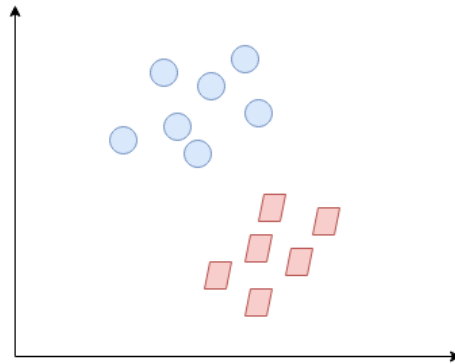
The training dataset for polynomial regression is non-linear in character.

To fit the intricate and non-linear functions and datasets, it employs a linear regression model. "In Polynomial regression, the original features are converted into Polynomial features of the desired degree (2,3,...,n) and then modeled using a linear model," explains the author. By applying this algorithm to study and investigate the covid-19 dataset it gives us the accuracy of about 99 percent which is very great but the only drawback that comes with this algorithm is that it can linearly increase but it will not be able to take good curves when the case will start decreasing all of a sudden. So in order to rise above it we will try another algorithm which is support vector regression

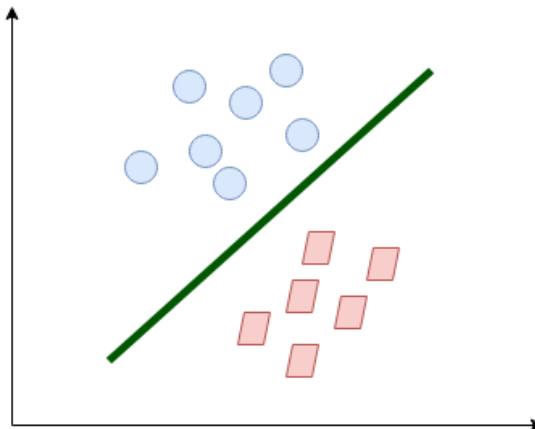
iii) Support Vector Regression

Support Vector Machines (SVM) are commonly employed in machine learning for categorization challenges.

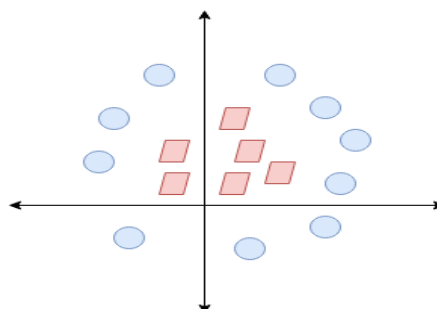
So, what is a Support Vector Machine (SVM) and how does it work? Let's start with a basic understanding of SVM. Assume we have a plot with two label classes, as illustrated in the diagram:



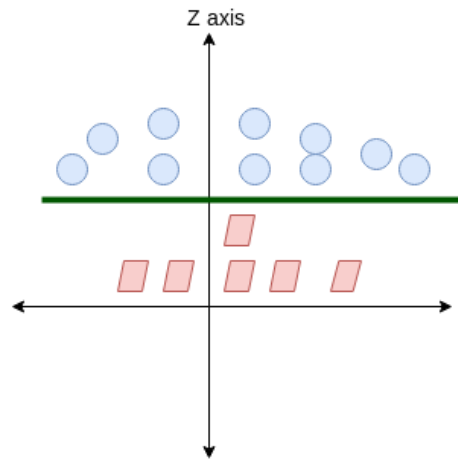
Can you decide where the dividing line will be drawn? You could have thought of this:



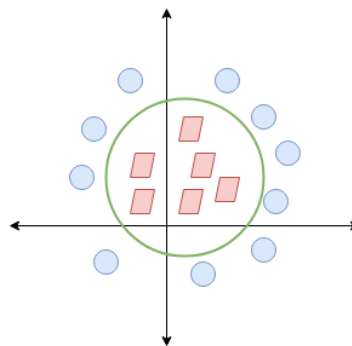
The line effectively divides the classes. Simple class separation is essentially what SVM accomplishes. Now, here's what the data looked like:



There isn't a straightforward line separating these two classes in this case. As a result, we'll expand our dimension and add a new dimension to the z-axis. These two classes can now be distinguished:



This line maps to the circular boundary when we transfer it back to the original plane, as shown here:



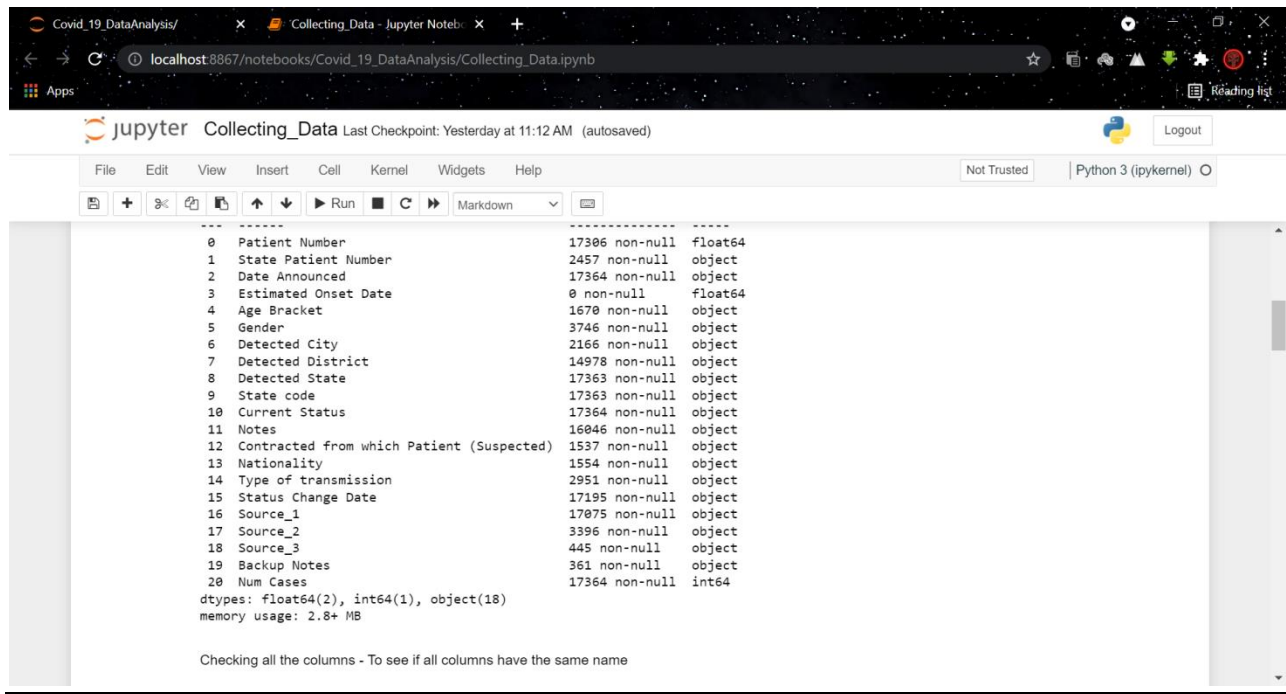
This is precisely what SVM accomplishes! It looks for a line or hyperplane (in multidimensional space) that divides these two classes. The new point is then classified based on whether it is on the positive or negative side of the hyperplane, as determined by the classes to be predicted.

Support Vector Regression (SVR) is a regression technique that uses the same principles as SVM. Let's take a few moments to grasp the concept of SVR. On the basis of a training sample, the aim of regression is to identify a function that approximates mapping from an input domain to real numbers. So let's take a closer look at how SVR truly works and by applying this algorithm we get a accuracy of 98 percent.

Merits of Proposed system:

- Forecast of growth in cases in future.
- Help in Better Prevention of upcoming situation.
- Getting the exact idea of current scenario using different representations (pie charts, graphs).
- Understanding the overall spread in India and the world by latitude and longitude.

Covid-19 Data Analysis



Covid_19_DataAnalysis/ x Collecting_Data - Jupyter Noteb: x +
localhost:8867/notebooks/Covid_19_DataAnalysis/Collecting_Data.ipynb

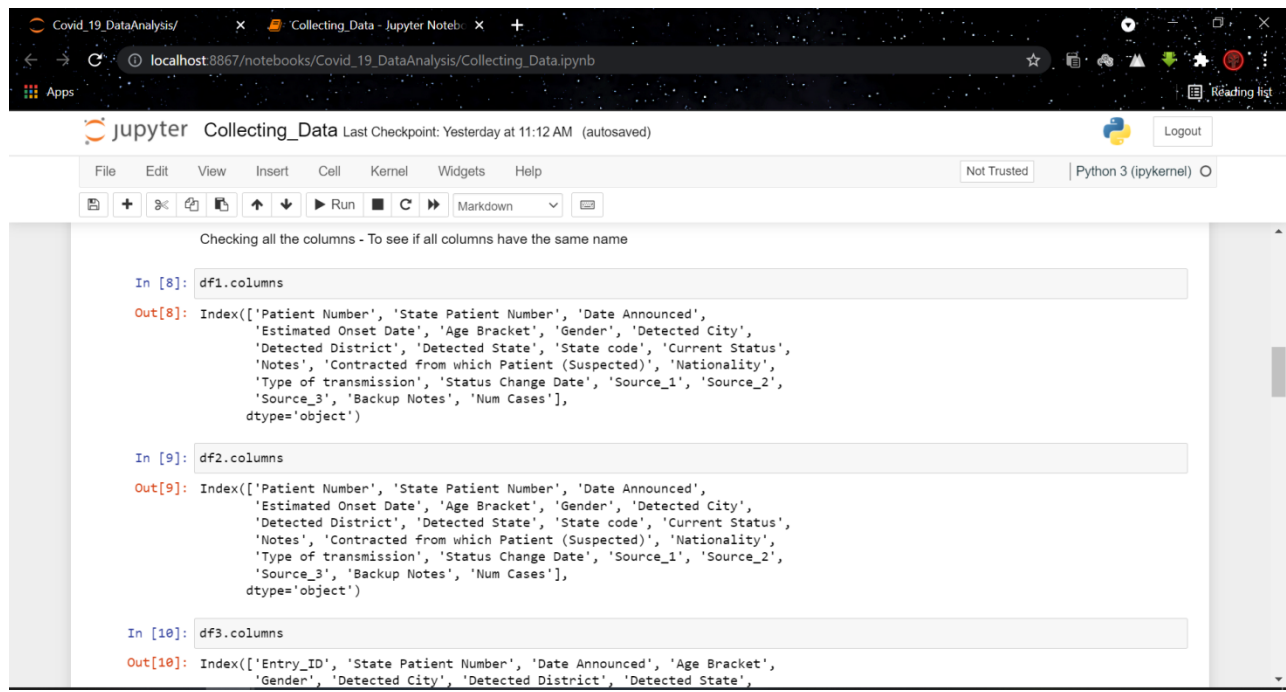
Jupyter Collecting_Data Last Checkpoint: Yesterday at 11:12 AM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
0 Patient Number 17306 non-null float64
1 State Patient Number 2457 non-null object
2 Date Announced 17364 non-null object
3 Estimated Onset Date 0 non-null float64
4 Age Bracket 1670 non-null object
5 Gender 3746 non-null object
6 Detected City 2166 non-null object
7 Detected District 14978 non-null object
8 Detected State 17363 non-null object
9 State code 17363 non-null object
10 Current Status 17364 non-null object
11 Notes 16046 non-null object
12 Contracted from which Patient (Suspected) 1537 non-null object
13 Nationality 1554 non-null object
14 Type of transmission 2951 non-null object
15 Status Change Date 17195 non-null object
16 Source_1 17075 non-null object
17 Source_2 3396 non-null object
18 Source_3 445 non-null object
19 Backup Notes 361 non-null object
20 Num Cases 17364 non-null int64

dtypes: float64(2), int64(1), object(18)
memory usage: 2.8+ MB
```

Checking all the columns - To see if all columns have the same name



Covid_19_DataAnalysis/ x Collecting_Data - Jupyter Noteb: x +
localhost:8867/notebooks/Covid_19_DataAnalysis/Collecting_Data.ipynb

Jupyter Collecting_Data Last Checkpoint: Yesterday at 11:12 AM (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
Checking all the columns - To see if all columns have the same name

In [8]: df1.columns
Out[8]: Index(['Patient Number', 'State Patient Number', 'Date Announced',
             'Estimated Onset Date', 'Age Bracket', 'Gender', 'Detected City',
             'Detected District', 'Detected State', 'State code', 'Current Status',
             'Notes', 'Contracted from which Patient (Suspected)', 'Nationality',
             'Type of transmission', 'Status Change Date', 'Source_1', 'Source_2',
             'Source_3', 'Backup Notes', 'Num Cases'],
            dtype='object')

In [9]: df2.columns
Out[9]: Index(['Patient Number', 'State Patient Number', 'Date Announced',
             'Estimated Onset Date', 'Age Bracket', 'Gender', 'Detected City',
             'Detected District', 'Detected State', 'State code', 'Current Status',
             'Notes', 'Contracted from which Patient (Suspected)', 'Nationality',
             'Type of transmission', 'Status Change Date', 'Source_1', 'Source_2',
             'Source_3', 'Backup Notes', 'Num Cases'],
            dtype='object')

In [10]: df3.columns
Out[10]: Index(['Entry_ID', 'State Patient Number', 'Date Announced', 'Age Bracket',
              'Gender', 'Detected City', 'Detected District', 'Detected State',
```

Covid-19 Data Analysis

This screenshot shows a Jupyter Notebook cell where the user has inspected the columns of a DataFrame and then selected specific columns to retain across multiple data frames. The output shows the list of columns and the code used to filter them.

```

In [10]: df3.columns
Out[10]: Index(['Entry_ID', 'State Patient Number', 'Date Announced', 'Age Bracket',
              'Gender', 'Detected City', 'Detected District', 'Detected State',
              'State code', 'Num Cases', 'Current Status',
              'Contracted from which Patient (Suspected)', 'Notes', 'Source_1',
              'Source_2', 'Source_3', 'Nationality', 'Type of transmission',
              'Status Change Date', 'Patient Number'],
              dtype='object')

Retain Necessary Columns

In [11]: df1 = df1.loc[:,['Num Cases','Date Announced','Age Bracket','Gender','Detected City','Detected District','Detected State','Cur
df2 = df2.loc[:,['Num Cases','Date Announced','Age Bracket','Gender','Detected City','Detected District','Detected State','Cur
df3 = df3.loc[:,['Num Cases','Date Announced','Age Bracket','Gender','Detected City','Detected District','Detected State','Cur
df4 = df4.loc[:,['Num Cases','Date Announced','Age Bracket','Gender','Detected City','Detected District','Detected State','Cur
df5 = df5.loc[:,['Num Cases','Date Announced','Age Bracket','Gender','Detected City','Detected District','Detected State','Cur
df6 = df6.loc[:,['Num Cases','Date Announced','Age Bracket','Gender','Detected City','Detected District','Detected State','Cur
df7 = df7.loc[:,['Num Cases','Date Announced','Age Bracket','Gender','Detected City','Detected District','Detected State','Cur
df8 = df8.loc[:,['Num Cases','Date Announced','Age Bracket','Gender','Detected City','Detected District','Detected State','Cur
  
```

This screenshot shows the final step of the data merging process. The user has appended all the filtered data frames into a single DataFrame. The output is a table with 10 columns: Num Cases, Date Announced, Age Bracket, Gender, Detected City, Detected District, Detected State, and Current Status. The table contains 10 rows of data, with some rows having missing values (NaN).

```

In [12]: df = df1.append([df2,df3,df4,df5,df6,df7,df8])
df
Out[12]:

```

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered
...
22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized
22791	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered
22792	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered
22793	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered
22794	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered

Covid-19 Data Analysis

Making separate columns for Day, Month, and Year

```
In [13]: DATE = df['Date Announced'].str.split('/', expand=True)
DATE.columns=['Day', 'Month', 'Year']
DATE
```

Out[13]:

	Day	Month	Year
0	30	01	2020
1	02	02	2020
2	03	02	2020
3	02	03	2020
4	02	03	2020
...
22790	07	07	2020
22791	07	07	2020
22792	07	07	2020
22793	07	07	2020
22794	07	07	2020

145849 rows x 3 columns

Concatenating both the data frames along axis=1

```
In [14]: df=pd.concat([df,DATE],axis=1)
df
```

Out[14]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	02	03	2020
...
22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	07	07	2020
22791	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22792	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22793	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	07	07	2020
22794	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered	07	07	2020

Covid-19 Data Analysis

Saving all the data in a single CSV file

```
In [15]: df.to_csv('Covid19India.csv')
```

FINAL DATASET

```
In [16]: df
```

Out[16]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	02	03	2020
...
22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	07	07	2020
22791	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22792	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020

Reading the CSV file that was already created previously

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

```
In [2]: df=pd.read_csv('Covid19India.csv',low_memory=False)
df
```

Out[2]:

	Unnamed: 0	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	1	2020
1	1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	2	2	2020
2	2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	3	2	2020
3	3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...
145844	22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	7	7	2020

Covid-19 Data Analysis

Removing unnamed column from the data

```
In [3]: data = df.iloc[:,1:]
data
```

Out[3]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	1	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	2	2	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	3	2	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...
145844	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	7	7	2020
145845	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145846	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145847	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	7	7	2020
145848	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered	7	7	2020

Inspecting the data frame

```
In [4]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145849 entries, 0 to 145848
Data columns (total 11 columns):
#   Column                Non-Null Count  Dtype
---  ---                ---
0   Num Cases              145846 non-null float64
1   Date Announced        145849 non-null object
2   Age Bracket            60013 non-null object
3   Gender                  62808 non-null object
4   Detected City           10949 non-null object
5   Detected District       137451 non-null object
6   Detected State          145840 non-null object
7   Current Status          145847 non-null object
8   Day                     145849 non-null int64
9   Month                   145849 non-null int64
10  Year                     145849 non-null int64
dtypes: float64(1), int64(3), object(7)
memory usage: 12.2+ MB
```

Covid-19 Data Analysis

Inspect Null values in each column and calculating the percentage of null values in each column.

```
In [5]: round(data.isnull().sum(axis=0).sort_values(ascending=True)/len(data)*100,2)
```

```
Out[5]: Date Announced      0.00  
Day                          0.00  
Month                        0.00  
Year                         0.00  
Current Status              0.00  
Num Cases                   0.00  
Detected State              0.01  
Detected District           5.76  
Gender                      56.94  
Age Bracket                 58.85  
Detected City               92.49  
dtype: float64
```

Total Covid-19 cases month-wise (This sum of monthly data consist of Hospitalized, Recovered, and Deaths) so we will group the data by only those rows where Current Status is Hospitalized.

```
In [6]: data.groupby('Month')['Num Cases'].sum()
```

```
Out[6]: Month  
1          1.0  
2          2.0  
3       1635.0  
4       36078.0  
5      242853.0  
6      663178.0  
7      270185.0  
Name: Num Cases, dtype: float64
```

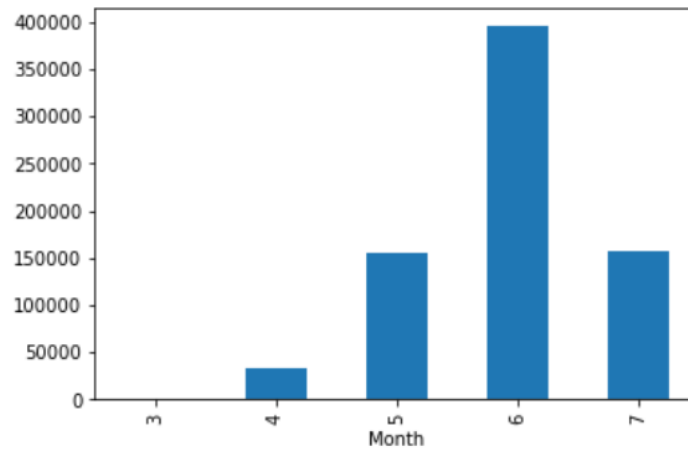
```
In [7]: M = data[data['Current Status']=='Hospitalized'].groupby('Month')['Num Cases'].sum()  
M
```

```
Out[7]: Month  
3         1431.0  
4        33209.0  
5        155781.0  
6        395144.0  
7         157701.0  
Name: Num Cases, dtype: float64
```

```
In [8]: M.plot.bar()  
plt.show()
```


Covid-19 Data Analysis

```
In [8]: M.plot.bar()  
plt.show()
```



The screenshot shows a Jupyter Notebook interface with the following content:

Total Male/Female affected by coronavirus

```
In [9]: data.groupby('Gender')['Num Cases'].sum()
```

```
Out[9]: Gender  
F          21294.0  
M          42795.0  
M           1.0  
Non-Binary  12.0  
Name: Num Cases, dtype: float64
```

What we can see here is that number of affected males is double than the no. of females affected. This means that males have double the chance of contracting covid-19 than a female.

Q) Which age group is infected the most?

For this, we will group the data according to age and then use the sum function to find the no. of cases for each age.

```
In [10]: data.groupby('Age Bracket')['Num Cases'].sum()
```

```
Out[10]: Age Bracket  
0.0     5.0  
0.1    18.0  
0.2     4.0  
0.25    1.0
```

Covid-19 Data Analysis

Q) Which age group is infected the most?

For this, we will group the data according to age and then use the sum function to find the no. of cases for each age.

```
In [10]: data.groupby('Age Bracket')['Num Cases'].sum()
```

```
Out[10]: Age Bracket
0.0      5.0
0.1     18.0
0.2      4.0
0.25     1.0
0.3      6.0
...
97       1.0
97.0     2.0
98.0     2.0
99       2.0
99.0     1.0
Name: Num Cases, Length: 224, dtype: float64
```

Now we will sort the above data in descending order to find the age with the highest no. of infected people.

```
In [11]: data.groupby('Age Bracket')['Num Cases'].sum().sort_values(ascending=False)
```

```
Out[11]: Age Bracket
30.0      1446.0
40.0      1242.0
35.0      1215.0
25.0      1198.0
32.0      1162.0
...
29.6        1.0
5 Months    1.0
5 months    1.0
54.9        1.0
99.0        1.0
Name: Num Cases, Length: 224, dtype: float64
```

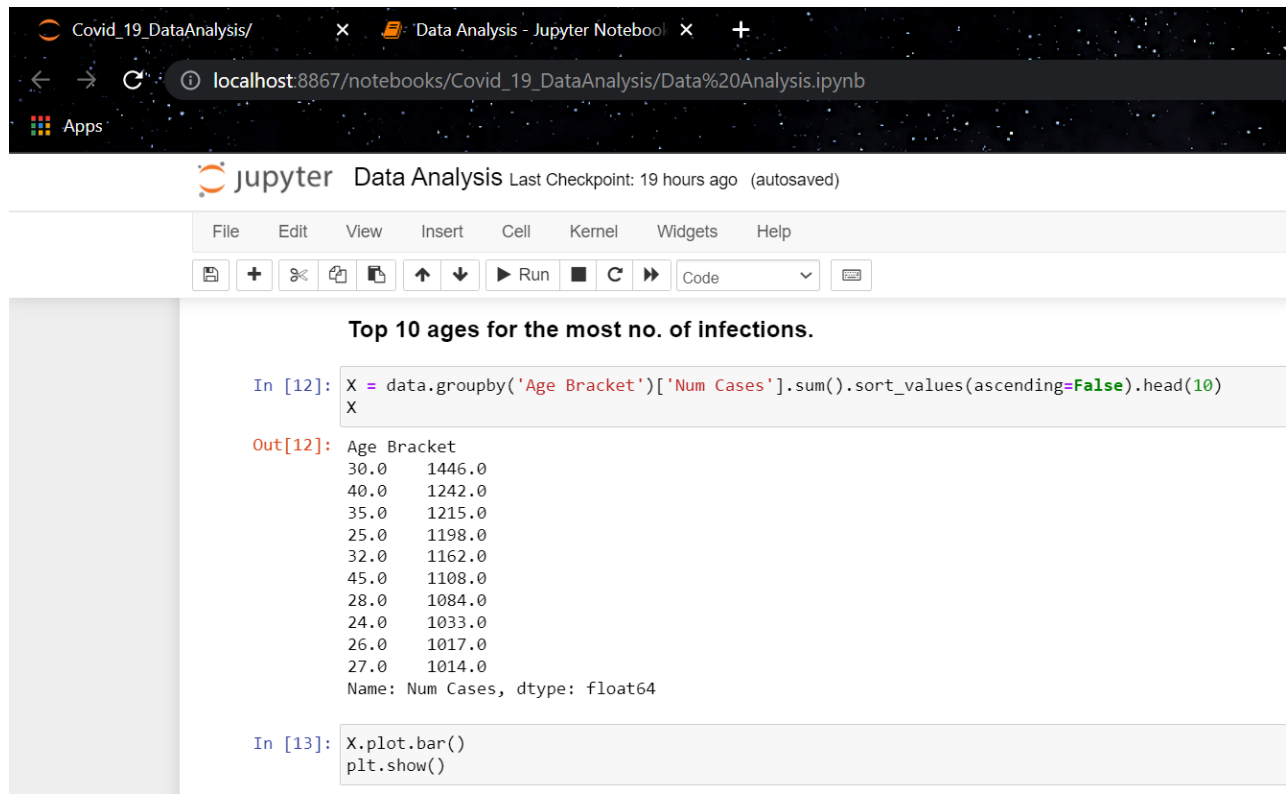
Now we will sort the above data in descending order to find the age with the highest no. of infected people.

```
In [11]: data.groupby('Age Bracket')['Num Cases'].sum().sort_values(ascending=False)
```

```
Out[11]: Age Bracket
30.0      1446.0
40.0      1242.0
35.0      1215.0
25.0      1198.0
32.0      1162.0
...
29.6        1.0
5 Months    1.0
5 months    1.0
54.9        1.0
99.0        1.0
Name: Num Cases, Length: 224, dtype: float64
```

The above data shows that the age=30 is the highest infected group with a total of 1446 infected people. So, the people of age=30 have the maximum chance of getting Covid-19.

Covid-19 Data Analysis

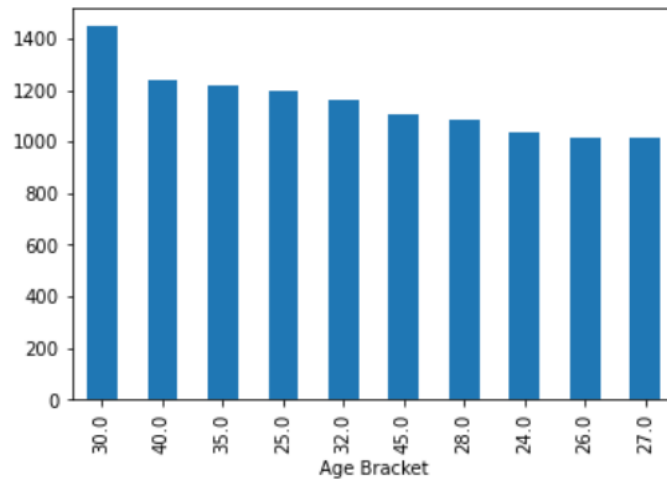


The screenshot shows a Jupyter Notebook window titled "Data Analysis - Jupyter Notebook" with the URL "localhost:8867/notebooks/Covid_19_DataAnalysis/Data%20Analysis.ipynb". The notebook interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for file operations and execution. The main content area displays the following code and output:

```
In [12]: X = data.groupby('Age Bracket')['Num Cases'].sum().sort_values(ascending=False).head(10)
X
```

```
Out[12]: Age Bracket
30.0    1446.0
40.0    1242.0
35.0    1215.0
25.0    1198.0
32.0    1162.0
45.0    1108.0
28.0    1084.0
24.0    1033.0
26.0    1017.0
27.0    1014.0
Name: Num Cases, dtype: float64
```

```
In [13]: X.plot.bar()
plt.show()
```



Covid-19 Data Analysis

Covid_19_DataAnalysis/ Data Analysis - Jupyter Notebook

localhost:8867/notebooks/Covid_19_DataAnalysis/Data%20Analysis.ipynb

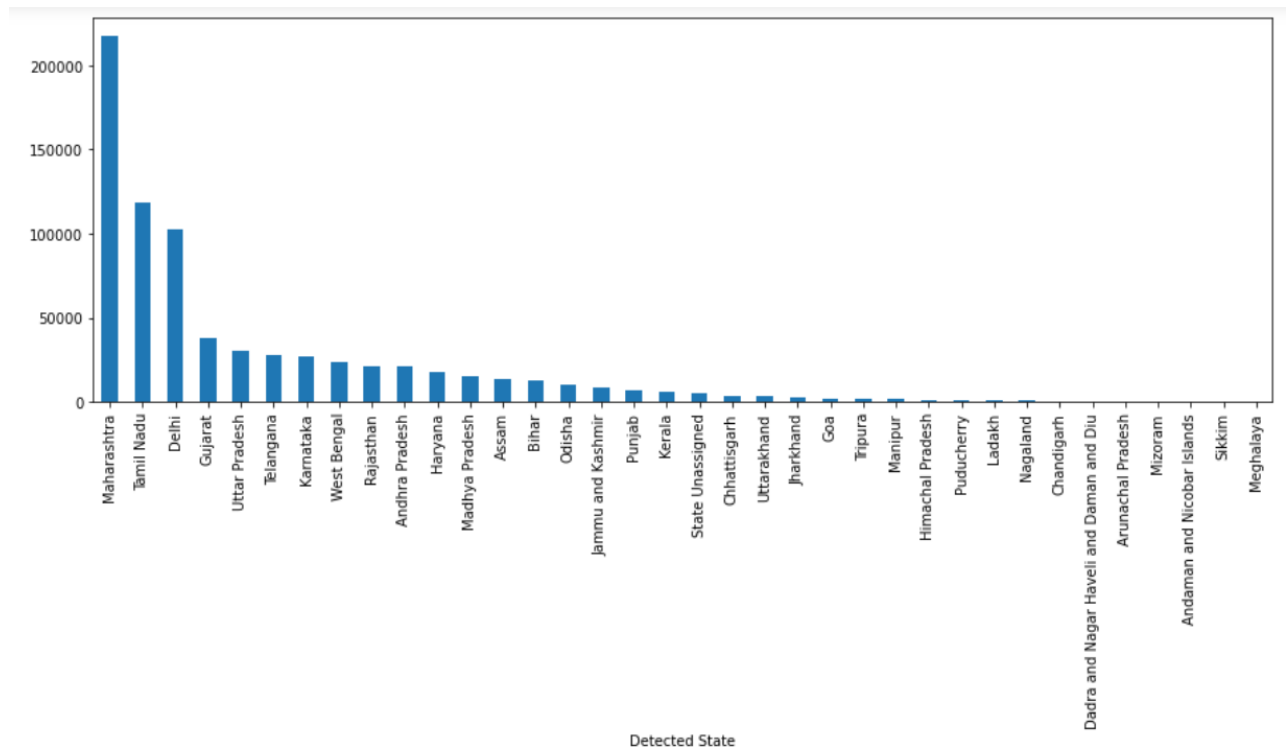
jupyter Data Analysis Last Checkpoint: 19 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Check State-Wise Total Cases in India

```
In [14]: Y=data[data['Current Status']=='Hospitalized'].groupby('Detected State')['Num Cases'].sum().sort_values(ascending=False)
```

```
Out[14]: Detected State
Maharashtra                217107.0
Tamil Nadu                 118587.0
Delhi                      102827.0
Gujarat                    37631.0
Uttar Pradesh              29959.0
Telangana                  27610.0
Karnataka                  26743.0
West Bengal                23831.0
Rajasthan                  21400.0
Andhra Pradesh             21195.0
Haryana                    17987.0
Madhya Pradesh             15625.0
Assam                      13337.0
Bihar                      12524.0
Odisha                     10096.0
Jammu and Kashmir          8930.0
Punjab                     6747.0
Kerala                     5834.0
State Unassigned           5034.0
Chhattisgarh               3407.0
Uttarakhand                3229.0
Jharkhand                  3018.0
Goa                         1903.0
Tripura                    1716.0
Manipur                    1429.0
```



Covid-19 Data Analysis

Covid_19_DataAnalysis/ x Data Analysis - Jupyter Notebook x +

localhost:8867/notebooks/Covid_19_DataAnalysis/Data%20Analysis.ipynb

Apps

jupyter Data Analysis Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Run Code

How many cases everyday?

```
In [16]: Day=data[data['Current Status']=='Hospitalized'].groupby(['Month','Day'])['Num Cases'].sum()
Day
```

Out[16]:

Month	Day	Num Cases
4	5	1.0
3	7	2.0
9	4	4.0
10	4	4.0
...
3	3	22718.0
4	4	24018.0
7	5	23942.0
6	6	22500.0
7	7	23147.0

124 rows x 1 columns

Covid_19_DataAnalysis/ x Data Analysis - Jupyter Notebook x +

localhost:8867/notebooks/Covid_19_DataAnalysis/Data%20Analysis.ipynb

Apps

jupyter Data Analysis Last Checkpoint: 19 hours ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help Not Trusted

Run Code

```
In [17]: Day.unstack(level=0).plot(kind='bar',subplots=True,figsize=(10,10))
plt.show()
```

(Num Cases, 3)

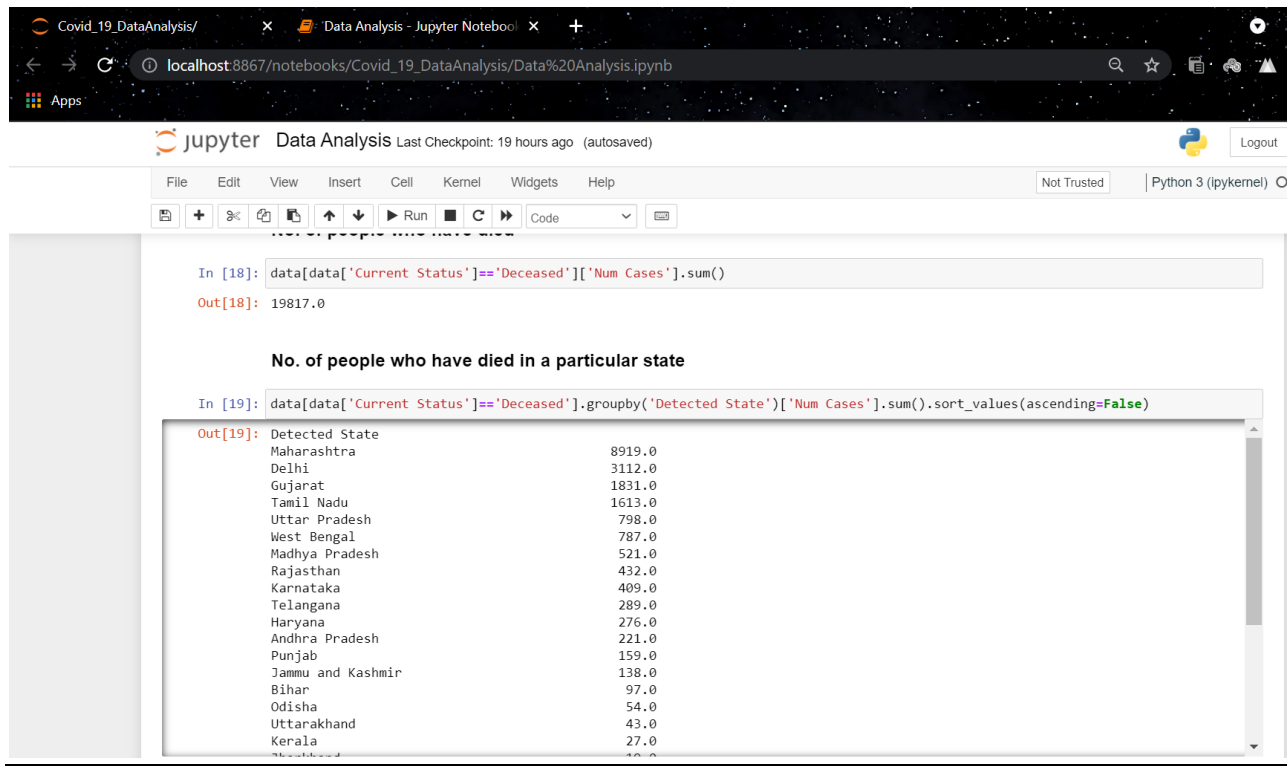
(Num Cases, 4)

(Num Cases, 5)

(Num Cases, 6)

(Num Cases, 7)

Covid-19 Data Analysis

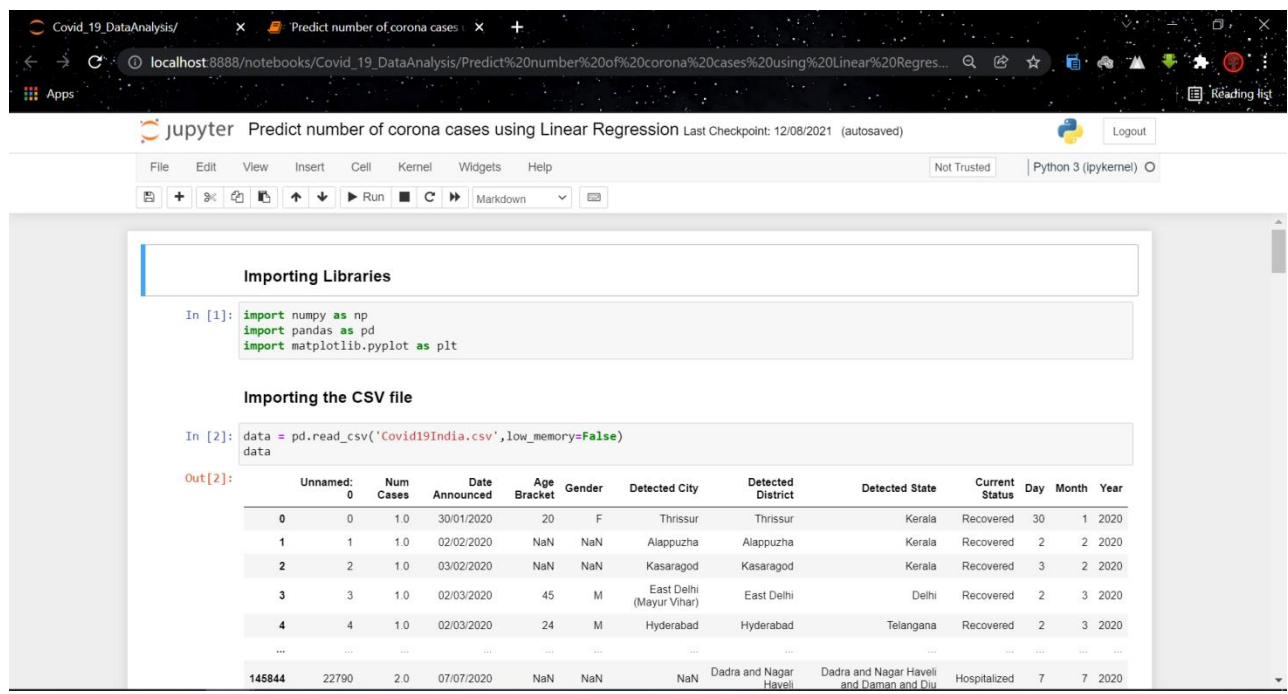


```
In [18]: data[data['Current Status']=='Deceased']['Num Cases'].sum()
Out[18]: 19817.0
```

No. of people who have died in a particular state

```
In [19]: data[data['Current Status']=='Deceased'].groupby('Detected State')['Num Cases'].sum().sort_values(ascending=False)
```

Detected State	Num Cases
Maharashtra	8919.0
Delhi	3112.0
Gujarat	1831.0
Tamil Nadu	1613.0
Uttar Pradesh	798.0
West Bengal	787.0
Madhya Pradesh	521.0
Rajasthan	432.0
Karnataka	409.0
Telangana	289.0
Haryana	276.0
Andhra Pradesh	221.0
Punjab	159.0
Jammu and Kashmir	138.0
Bihar	97.0
Odisha	54.0
Uttarakhand	43.0
Kerala	27.0



```
In [1]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

Importing the CSV file

```
In [2]: data = pd.read_csv('Covid19India.csv', low_memory=False)
data
```

Unnamed: 0	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year	
0	0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	1	2020
1	1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	2	2	2020
2	2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	3	2	2020
3	3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...
145844	22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	7	7	2020

Covid-19 Data Analysis

Removing the unnecessary columns

```
In [3]: data.iloc[:,1:]
```

Out[3]:

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	1	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	2	2	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	3	2	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...
145844	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	7	7	2020
145845	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145846	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145847	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	7	7	2020
145848	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered	7	7	2020

145849 rows x 11 columns

Grouping data day-wise (for applying Linear Regression)

Grouping data day-wise (for applying Linear Regression)

```
In [4]: Day = data[data['Current Status']=='Hospitalized'].groupby(['Month', 'Day'])[['Num Cases']].sum()
Day
```

Out[4]:

Month	Day	Num Cases
4	5.0	5.0
5	1.0	1.0
3	7	2.0
9	4.0	4.0
10	4.0	4.0
...
3	3	22718.0
4	4	24018.0
7	5	23942.0
6	6	22500.0
7	7	23147.0

124 rows x 1 columns

Covid-19 Data Analysis

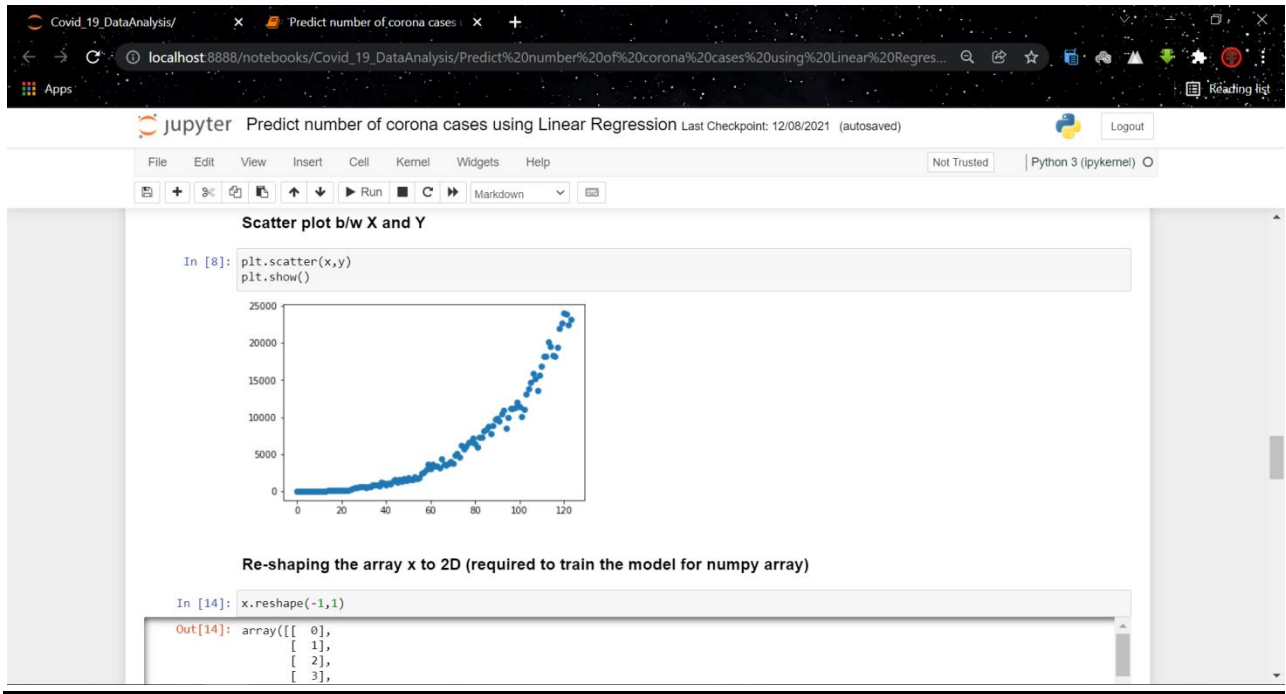
This screenshot shows a Jupyter Notebook interface with the following content:

- Kernel: Python 3 (ipykernel)
- Cell 7: `23147.0`
- Text: 124 rows x 1 columns
- Section: **Generating the value of X-axis variable (no. of days)**
- Code: `In [5]: len(Day)`
- Output: `Out[5]: 124`
- Section: **Generating an array equal to the length of Day**
- Code: `In [6]: x = np.arange(len(Day))`
- Output: `Out[6]: array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123])`

This screenshot shows the continuation of the Jupyter Notebook with the following content:

- Code: `In [7]: y = Day.values`
- Output: `Out[7]: array([[5.0000e+00], [1.0000e+00], [2.0000e+00], [4.0000e+00], [4.0000e+00], [8.0000e+00], [4.0000e+00], [6.0000e+00], [1.1000e+01], [8.0000e+00], [1.2000e+01], [1.4000e+01], [2.2000e+01], [2.1000e+01], [5.2000e+01], [6.7000e+01], [5.9000e+01], [8.2000e+01], [6.3000e+01], [7.6000e+01]])`
- Section: **Scatter plot b/w X and Y**
- Code: `In [8]: plt.scatter(x,y)`
`plt.show()`

Covid-19 Data Analysis



Re-shaping the array x to 2D (required to train the model for numpy array)

```
In [14]: x.reshape(-1,1)
Out[14]: array([[ 0],
 [ 1],
 [ 2],
 [ 3],
 [ 4],
 [ 5],
 [ 6],
 [ 7],
 [ 8],
 [ 9],
 [10],
 [11],
 [12],
 [13],
 [14],
 [15],
 [16],
 [17],
 [18],
 [19],
 [20],
 [21],
 [22],
 [23],
 [24],
 [25],
 [26],
 [27],
 [28],
 [29],
 [30],
 [31],
 [32],
 [33],
 [34],
 [35],
 [36],
 [37],
 [38],
 [39],
 [40],
 [41],
 [42],
 [43],
 [44],
 [45],
 [46],
 [47],
 [48],
 [49],
 [50],
 [51],
 [52],
 [53],
 [54],
 [55],
 [56],
 [57],
 [58],
 [59],
 [60],
 [61],
 [62],
 [63],
 [64],
 [65],
 [66],
 [67],
 [68],
 [69],
 [70],
 [71],
 [72],
 [73],
 [74],
 [75],
 [76],
 [77],
 [78],
 [79],
 [80],
 [81],
 [82],
 [83],
 [84],
 [85],
 [86],
 [87],
 [88],
 [89],
 [90],
 [91],
 [92],
 [93],
 [94],
 [95],
 [96],
 [97],
 [98],
 [99],
 [100],
 [101],
 [102],
 [103],
 [104],
 [105],
 [106],
 [107],
 [108],
 [109],
 [110],
 [111],
 [112],
 [113],
 [114],
 [115],
 [116],
 [117],
 [118],
 [119],
 [120]])
```

Importing Linear Regression model from Scikit-learn library

```
In [1]: from sklearn.linear_model import LinearRegression
regressor = LinearRegression()
```


Covid-19 Data Analysis

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [113]: regressor.intercept_ #c
Out[113]: array([-4553.47612903])

In [115]: regressor.coef_ # m
Out[115]: array([[171.5049882]])

In [116]: regressor.predict([[152]])
Out[116]: array([[21515.2820771]])

In [118]: df
Out[118]:
```

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	02	03	2020
...
22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	07	07	2020

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [119]: Day = df[df["Current Status"]=="Hospitalized"].groupby(["Month","Day"])["Num Cases"].sum()
Day
Out[119]:
```

Month	Day	Num Cases
04	05	5.0
05	01	1.0
03	07	2.0
09	04	4.0
10	04	4.0
...
03	03	22718.0
04	04	24018.0
07	05	23942.0
06	06	22500.0
07	07	23147.0

124 rows x 1 columns

```
In [120]: x = np.arange(len(Day))
x
Out[120]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12])
```

Covid-19 Data Analysis

localhost:8890/notebooks/Covid_19_DataAnalysis/covid-19%20forecasting.ipynb

jupyter covid-19 forecasting (autosaved) Logout

```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
```

In [120]: `x = np.arange(len(Day))`
`x`

```
Out[120]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118, 119, 120, 121, 122, 123])
```

In [121]: `y = Day.values`
`y`

```
[1.4740e+04],
[1.5918e+04],
[1.5151e+04],
[1.3560e+04],
[1.5656e+04],
[1.6868e+04],
[1.8205e+04],
[1.8255e+04],
[2.0142e+04],
[1.9610e+04],
[1.8339e+04],
[1.8256e+04],
[1.9429e+04],
```

localhost:8890/notebooks/Covid_19_DataAnalysis/covid-19%20forecasting.ipynb

jupyter covid-19 forecasting (autosaved) Logout

```
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)
```

In [123]: `x = x.reshape(-1,1)`

In [144]: `from sklearn.preprocessing import PolynomialFeatures`
`poly = PolynomialFeatures(degree = 5)`
`X = poly.fit_transform(x)`
`X`

```
Out[144]: array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
0.00000000e+00, 0.00000000e+00],
[1.00000000e+00, 1.00000000e+00, 1.00000000e+00, 1.00000000e+00, 1.00000000e+00,
1.00000000e+00, 1.00000000e+00],
[1.00000000e+00, 2.00000000e+00, 4.00000000e+00, 8.00000000e+00, 1.60000000e+01,
3.20000000e+01],
[1.00000000e+00, 3.00000000e+00, 9.00000000e+00, 2.70000000e+01, 8.10000000e+01,
2.43000000e+02],
[1.00000000e+00, 4.00000000e+00, 1.60000000e+01, 6.40000000e+01, 2.56000000e+02,
1.02400000e+03],
[1.00000000e+00, 5.00000000e+00, 2.50000000e+01, 1.25000000e+02, 6.25000000e+02,
3.12500000e+03],
[1.00000000e+00, 6.00000000e+00, 3.60000000e+01, 2.16000000e+02, 1.29600000e+03,
7.77600000e+03],
[1.00000000e+00, 7.00000000e+00, 4.90000000e+01, 3.43000000e+02, 2.40100000e+03,
1.68070000e+04],
[1.00000000e+00, 8.00000000e+00, 6.40000000e+01, 5.12000000e+02, 4.09600000e+03,
3.27680000e+04],
[1.00000000e+00, 9.00000000e+00, 8.10000000e+01, 7.29000000e+02,
```

In [145]: `pd.DataFrame(X)`

```
Out[145]:
```

Covid-19 Data Analysis

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [145]: pd.DataFrame(X)
```

	0	1	2	3	4	5
0	1.0	0.0	0.0	0.0	0.0	0.000000e+00
1	1.0	1.0	1.0	1.0	1.0	1.000000e+00
2	1.0	2.0	4.0	8.0	16.0	3.200000e+01
3	1.0	3.0	9.0	27.0	81.0	2.430000e+02
4	1.0	4.0	16.0	64.0	256.0	1.024000e+03
...
119	1.0	119.0	14161.0	1685159.0	200533921.0	2.386354e+10
120	1.0	120.0	14400.0	1728000.0	207360000.0	2.488320e+10
121	1.0	121.0	14641.0	1771561.0	214358881.0	2.593742e+10
122	1.0	122.0	14884.0	1815848.0	221533456.0	2.702708e+10
123	1.0	123.0	15129.0	1860867.0	228886641.0	2.815306e+10

124 rows x 6 columns

```
In [146]: from sklearn.linear_model import LinearRegression  
regressor = LinearRegression()  
regressor.fit(X,y)
```

```
Out[146]: LinearRegression()
```

The screenshot shows the continuation of the Jupyter Notebook with the following content:

```
In [146]: from sklearn.linear_model import LinearRegression  
regressor = LinearRegression()  
regressor.fit(X,y)
```

```
Out[146]: LinearRegression()
```

```
In [147]: regressor.intercept_ #c
```

```
Out[147]: array([-111.29209183])
```

```
In [148]: regressor.coef_ #m
```

```
Out[148]: array([[ 0.00000000e+00,  3.47987675e+01, -2.33294348e+00,  
                7.91412887e-02, -7.69989889e-04,  3.00699692e-06]])
```

```
In [149]: yp = regressor.predict(X)  
yp
```

```
Out[149]: array([[ -1.11292092e+02],  
                [ -7.87478936e+01],  
                [ -5.04054241e+01],  
                [ -2.58171045e+01],  
                [ -4.55311345e+00],  
                [  1.37989727e+01],  
                [  2.96345415e+01],  
                [  4.33323048e+01],  
                [  5.52546596e+01],  
                [  6.57480494e+01],  
                [  7.51433243e+01],  
                [  8.37561024e+01],  
                [  9.18871302e+01],
```

Covid-19 Data Analysis

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [150]: plt.scatter(x,y)
plt.plot(x,y,color="k")
plt.show()
```

```
In [151]: regressor.score(X,y)*100
Out[151]: 99.04926030995705
```

```
In [152]: x
Out[152]: array([[ 0],
 [ 1],
 [ 2],
 [ 3],
 [ 4]])
```

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [156]: regressor.predict(poly.transform([[125]]))
Out[156]: array([[26139.59234236]])
```

```
In [ ]:
```

```
In [158]: df
Out[158]:
```

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	02	03	2020
...
22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	07	07	2020
22791	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22792	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22793	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	07	07	2020
22794	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered	07	07	2020

Covid-19 Data Analysis

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [159]: Day = df[df["Current Status"]=="Hospitalized"].groupby(["Month","Day"])["Num Cases"].sum()
Day
Out[159]:
```

Month	Day	Num Cases
04	05	5.0
05	05	1.0
03	07	2.0
09	07	4.0
10	07	4.0
...
03	03	22718.0
04	03	24018.0
07	05	23942.0
06	06	22500.0
07	07	23147.0

124 rows x 1 columns

```
In [160]: x = np.arange(len(Day))
x
Out[160]: array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12,
          13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25,
```

The screenshot shows the continuation of the Jupyter Notebook with the following code and output:

```
91, 92, 93, 94, 95, 96, 97, 98, 99, 100, 101, 102, 103,
104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114, 115, 116,
117, 118, 119, 120, 121, 122, 123})

In [161]: y = Day.values
-y
Out[161]: array([[5.0000e+00],
 [1.0000e+00],
 [2.0000e+00],
 [4.0000e+00],
 [4.0000e+00],
 [8.0000e+00],
 [4.0000e+00],
 [6.0000e+00],
 [1.1000e+01],
 [8.0000e+00],
 [1.2000e+01],
 [1.4000e+01],
 [2.2000e+01],
 [2.1000e+01],
 [5.2000e+01],
 [6.7000e+01],
 [5.9000e+01],
 [8.2000e+01],
 [6.3000e+01],
 ...])

In [163]: x = x.reshape(-1,1)
y = y.reshape(-1,1)
```

Covid-19 Data Analysis

The screenshot shows a Jupyter Notebook interface for 'covid-19 forecasting'. The browser address bar indicates the URL is localhost:8890/notebooks/Covid_19_DataAnalysis/covid-19%20forecasting.ipynb. The notebook title is 'jupyter covid-19 forecasting (autosaved)'. The kernel is 'Python 3 (pykernel)'. The code in cell [164] imports StandardScaler and applies it to data X and y. Cell [171] prints the transformed data Sx.

```
In [164]: from sklearn.preprocessing import StandardScaler
sc_X = StandardScaler()
sc_y = StandardScaler()
Sx = sc_X.fit_transform(x)
Sy = sc_y.fit_transform(y)
```

```
In [171]: Sx
```

```
Out[171]: array([[ -1.71813853],
[-1.69020132],
[-1.6622641 ],
[-1.63432689],
[-1.60638968],
[-1.57845247],
[-1.55051526],
[-1.52257804],
[-1.49464083],
[-1.46670362],
[-1.43876641],
[-1.4108292 ],
[-1.38289199],
[-1.35495479],
[-1.32701756],
[-1.29908035],
[-1.27114314],
[-1.24320593],
[-1.21526871],
[-1.1873315 ]])
```

```
In [166]: from sklearn.svm import SVR
```

The screenshot shows the continuation of the Jupyter Notebook. Cell [166] imports SVR and fits a model with an RBF kernel. Cell [167] calculates the score of the model. Cell [170] creates a scatter plot of the data with a fitted SVR curve.

```
In [166]: from sklearn.svm import SVR
regressor = SVR(kernel="rbf")
regressor.fit(Sx,Sy.ravel())
```

```
Out[166]: SVR()
```

```
In [167]: regressor.score(Sx,Sy)*100
```

```
Out[167]: 98.55266794938339
```

```
In [170]: plt.scatter(Sx,Sy)
plt.plot(Sx,regressor.predict(Sx),color="k",linewidth=3)
plt.show()
```

The plot displays a scatter of blue data points and a thick black fitted curve. The x-axis ranges from -1.5 to 1.5, and the y-axis ranges from -1.0 to 2.5. The curve shows a non-linear, upward-curving relationship between the variables.

Covid-19 Data Analysis

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [172]: df
```

	Num Cases	Date Announced	Age Bracket	Gender	Detected City	Detected District	Detected State	Current Status	Day	Month	Year
0	1.0	30/01/2020	20	F	Thrissur	Thrissur	Kerala	Recovered	30	01	2020
1	1.0	02/02/2020	NaN	NaN	Alappuzha	Alappuzha	Kerala	Recovered	02	02	2020
2	1.0	03/02/2020	NaN	NaN	Kasaragod	Kasaragod	Kerala	Recovered	03	02	2020
3	1.0	02/03/2020	45	M	East Delhi (Mayur Vihar)	East Delhi	Delhi	Recovered	02	03	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	02	03	2020
...
22790	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	07	07	2020
22791	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22792	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22793	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	07	07	2020
22794	1.0	07/07/2020	NaN	NaN	NaN	Wayanad	Kerala	Recovered	07	07	2020

145849 rows x 11 columns

```
In [173]: Day = df[df["Current Status"]=="Hospitalized"].groupby(["Month", "Day"])["Num Cases"].sum()
Day
```

Month	Day	Num Cases
04	05	1.0
03	07	2.0
09	10	4.0
10	10	4.0
...
03	03	22718.0
04	03	24018.0
07	05	23942.0
06	06	22500.0
07	07	23147.0

124 rows x 1 columns

The screenshot shows a Jupyter Notebook interface with the following content:

```
In [173]: Day = df[df["Current Status"]=="Hospitalized"].groupby(["Month", "Day"])["Num Cases"].sum()
Day
```

Month	Day	Num Cases
04	05	1.0
03	07	2.0
09	10	4.0
10	10	4.0
...
03	03	22718.0
04	03	24018.0
07	05	23942.0
06	06	22500.0
07	07	23147.0

124 rows x 1 columns

```
In [174]: x = np.arange(len(Day))
x = x.reshape(-1,1)
x
```

Day
04

Covid-19 Data Analysis

```

covid-19 forecasting - Jupyter No. x +
localhost:8890/notebooks/Covid_19_DataAnalysis/covid-19%20forecasting.ipynb

jupyter covid-19 forecasting (autosaved) Python 3 (pykernel)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (pykernel)

In [174]: x = np.arange(len(Day))
x = x.reshape(-1,1)
x

Out[174]: array([[ 0],
 [ 1],
 [ 2],
 [ 3],
 [ 4],
 [ 5],
 [ 6],
 [ 7],
 [ 8],
 [ 9],
 [10],
 [11],
 [12],
 [13],
 [14],
 [15],
 [16],
 [17],
 [18],
 [19]]

In [175]: y = Day.values
y

Out[175]: array([[5.0000e+00],
 [1.0000e+00],
 [2.0000e+00],
 [4.0000e+00],
 [6.0000e+00],
 [8.0000e+00],
 [1.0000e+01],
 [1.2000e+01],
 [1.4000e+01],
 [1.6000e+01],
 [1.8000e+01],
 [2.0000e+01],
 [2.2000e+01],
 [2.4000e+01],
 [2.6000e+01],
 [2.8000e+01],
 [3.0000e+01],
 [3.2000e+01],
 [3.4000e+01],
 [3.6000e+01],
 [3.8000e+01],
 [4.0000e+01],
 [4.2000e+01],
 [4.4000e+01],
 [4.6000e+01],
 [4.8000e+01],
 [5.0000e+01],
 [5.2000e+01],
 [5.4000e+01],
 [5.6000e+01],
 [5.8000e+01],
 [6.0000e+01],
 [6.2000e+01],
 [6.4000e+01],
 [6.6000e+01],
 [6.8000e+01],
 [7.0000e+01],
 [7.2000e+01],
 [7.4000e+01],
 [7.6000e+01],
 [7.8000e+01],
 [8.0000e+01],
 [8.2000e+01],
 [8.4000e+01],
 [8.6000e+01],
 [8.8000e+01],
 [9.0000e+01],
 [9.2000e+01],
 [9.4000e+01],
 [9.6000e+01],
 [9.8000e+01],
 [1.0000e+02],
 [1.0200e+02],
 [1.0400e+02],
 [1.0600e+02],
 [1.0800e+02],
 [1.1000e+02],
 [1.1200e+02],
 [1.1400e+02],
 [1.1600e+02],
 [1.1800e+02],
 [1.2000e+02],
 [1.2200e+02],
 [1.2400e+02],
 [1.2600e+02],
 [1.2800e+02],
 [1.3000e+02],
 [1.3200e+02],
 [1.3400e+02],
 [1.3600e+02],
 [1.3800e+02],
 [1.4000e+02],
 [1.4200e+02],
 [1.4400e+02],
 [1.4600e+02],
 [1.4800e+02],
 [1.5000e+02],
 [1.5200e+02],
 [1.5400e+02],
 [1.5600e+02],
 [1.5800e+02],
 [1.6000e+02],
 [1.6200e+02],
 [1.6400e+02],
 [1.6600e+02],
 [1.6800e+02],
 [1.7000e+02],
 [1.7200e+02],
 [1.7400e+02],
 [1.7600e+02],
 [1.7800e+02],
 [1.8000e+02],
 [1.8200e+02],
 [1.8400e+02],
 [1.8600e+02],
 [1.8800e+02],
 [1.9000e+02],
 [1.9200e+02],
 [1.9400e+02],
 [1.9600e+02],
 [1.9800e+02],
 [2.0000e+02]]
    
```

```

covid-19 forecasting - Jupyter No. x +
localhost:8890/notebooks/Covid_19_DataAnalysis/covid-19%20forecasting.ipynb

jupyter covid-19 forecasting (autosaved) Python 3 (pykernel)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (pykernel)

In [176]: from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(x,y)

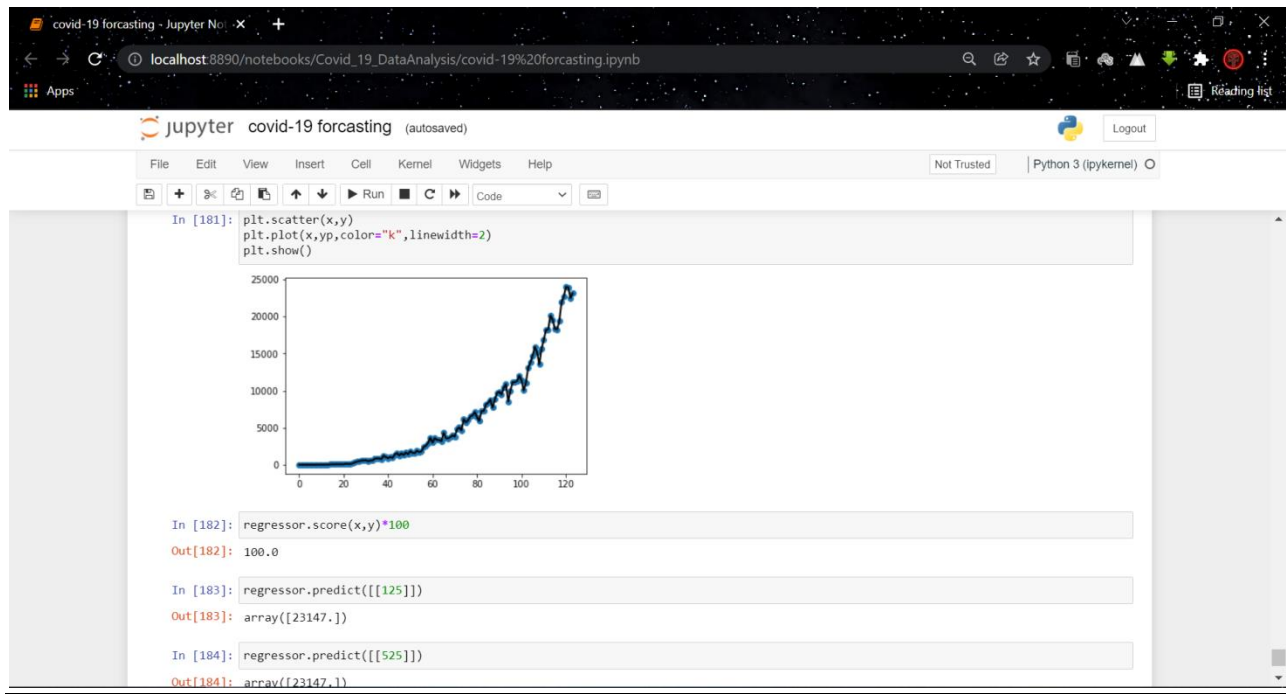
Out[176]: DecisionTreeRegressor()

In [178]: # here we dont have any coeff and intercept because its not a line

In [179]: yp = regressor.predict(x)
yp

Out[179]: array([[5.0000e+00, 1.0000e+00, 2.0000e+00, 4.0000e+00, 4.0000e+00,
 8.0000e+00, 4.0000e+00, 6.0000e+00, 1.1000e+01, 8.0000e+00,
 1.2000e+01, 1.4000e+01, 2.2000e+01, 2.1000e+01, 5.2000e+01,
 6.7000e+01, 5.9000e+01, 8.2000e+01, 6.3000e+01, 7.5000e+01,
 5.8000e+01, 1.4000e+02, 1.2300e+02, 1.0600e+02, 1.7800e+02,
 3.0600e+02, 4.2300e+02, 4.8500e+02, 5.5600e+02, 5.7600e+02,
 6.0600e+02, 4.8500e+02, 5.7000e+02, 5.6300e+02, 8.1200e+02,
 8.7000e+02, 8.5300e+02, 7.5800e+02, 1.2430e+03, 1.0310e+03,
 8.8400e+02, 1.0610e+03, 9.2200e+02, 1.3700e+03, 1.5790e+03,
 1.2390e+03, 1.5370e+03, 1.2920e+03, 1.6670e+03, 1.4080e+03,
 1.8350e+03, 1.6070e+03, 1.5680e+03, 1.9020e+03, 1.7050e+03,
 1.8020e+03, 2.3960e+03, 2.5640e+03, 2.9520e+03, 3.6560e+03,
 2.9710e+03, 3.6020e+03, 3.3440e+03, 3.3390e+03, 3.1750e+03,
 4.3110e+03, 3.5920e+03, 3.5620e+03, 3.7260e+03, 3.9910e+03,
 3.8080e+03, 4.7940e+03, 5.0490e+03, 4.6280e+03, 6.1540e+03,
 5.7200e+03, 6.0230e+03, 6.5360e+03, 6.6650e+03, 7.1110e+03,
 6.4140e+03, 5.9070e+03, 7.2460e+03, 7.2540e+03, 8.1380e+03,
 8.3640e+03, 8.7890e+03, 7.7230e+03, 8.8120e+03, 9.6890e+03,
 9.8470e+03, 9.4730e+03, 1.0400e+04, 1.0007e+04, 9.5360e+03,
 9.7110e+03, 9.8110e+03, 9.9110e+03, 1.0011e+04, 1.0111e+04, 1.0211e+04,
 1.0311e+04, 1.0411e+04, 1.0511e+04, 1.0611e+04, 1.0711e+04,
 1.0811e+04, 1.0911e+04, 1.1011e+04, 1.1111e+04, 1.1211e+04,
 1.1311e+04, 1.1411e+04, 1.1511e+04, 1.1611e+04, 1.1711e+04,
 1.1811e+04, 1.1911e+04, 1.2011e+04, 1.2111e+04, 1.2211e+04,
 1.2311e+04, 1.2411e+04, 1.2511e+04, 1.2611e+04, 1.2711e+04,
 1.2811e+04, 1.2911e+04, 1.3011e+04, 1.3111e+04, 1.3211e+04,
 1.3311e+04, 1.3411e+04, 1.3511e+04, 1.3611e+04, 1.3711e+04,
 1.3811e+04, 1.3911e+04, 1.4011e+04, 1.4111e+04, 1.4211e+04,
 1.4311e+04, 1.4411e+04, 1.4511e+04, 1.4611e+04, 1.4711e+04,
 1.4811e+04, 1.4911e+04, 1.5011e+04, 1.5111e+04, 1.5211e+04,
 1.5311e+04, 1.5411e+04, 1.5511e+04, 1.5611e+04, 1.5711e+04,
 1.5811e+04, 1.5911e+04, 1.6011e+04, 1.6111e+04, 1.6211e+04,
 1.6311e+04, 1.6411e+04, 1.6511e+04, 1.6611e+04, 1.6711e+04,
 1.6811e+04, 1.6911e+04, 1.7011e+04, 1.7111e+04, 1.7211e+04,
 1.7311e+04, 1.7411e+04, 1.7511e+04, 1.7611e+04, 1.7711e+04,
 1.7811e+04, 1.7911e+04, 1.8011e+04, 1.8111e+04, 1.8211e+04,
 1.8311e+04, 1.8411e+04, 1.8511e+04, 1.8611e+04, 1.8711e+04,
 1.8811e+04, 1.8911e+04, 1.9011e+04, 1.9111e+04, 1.9211e+04,
 1.9311e+04, 1.9411e+04, 1.9511e+04, 1.9611e+04, 1.9711e+04,
 1.9811e+04, 1.9911e+04, 2.0011e+04]]
    
```

Covid-19 Data Analysis



Limitations

“Data is not reality it only tries to approximate reality.”

- Every COVID cases may not be reported, checked and confirmed.
- Lag due to reporting or data entry.
- Actual v/s predicted plot may not be linear.(underestimate or overestimate)

Covid-19 data limitations are a challenging factor in predicting time series data.

Extension of Recurrent Neural Network (RNN) as a Short-Term Memory (LSTM) cell and its variants such as Stacked LSTM, Bi-directional LSTM and Convolutional LSTM are used to measure predictions.

Future Scope of the Project

This paperwork can be extended to higher levels in the future.

In the future, we can study the loss of the total economy at the end of Covid-19 in various regions and formulate a reasonable plan to recover it, to help countries recover the economy rate.

And the aerosol transmission of Covid-19 can be verified.

In addition the analysis of future prediction can be extended resulting in more accurate prediction to estimate a more accurate number of total cases in India.

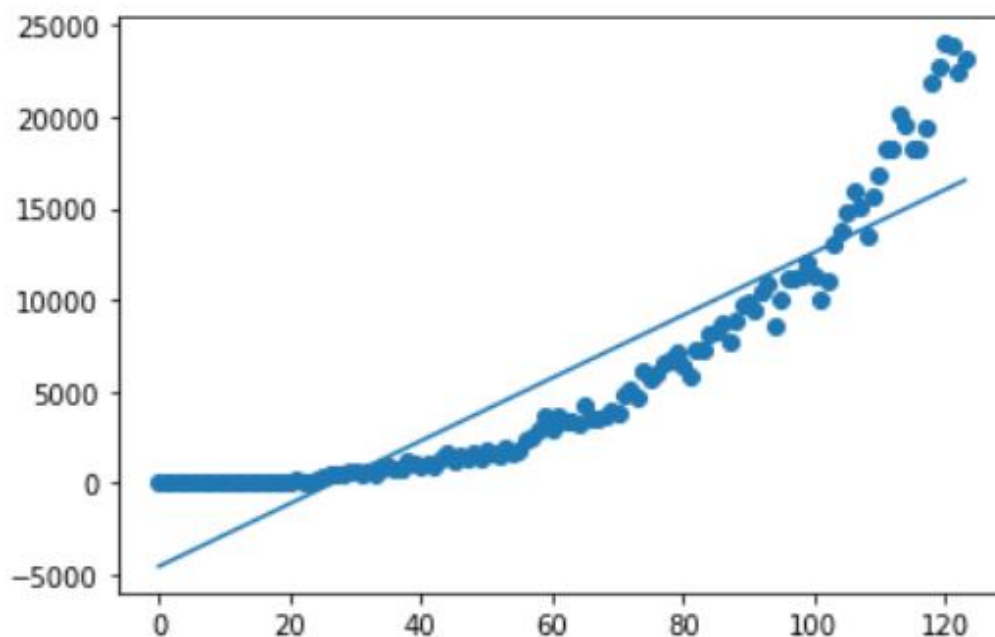
Results and Discussion

The main objective of the Project is to study and analyze COVID-19 prevalence. The spread of the disease in India compared to other countries, state-wise trend of the epidemic to know how it is spreading and to analyze the healthcare sector of India and finally to predict the future of the epidemic in India.

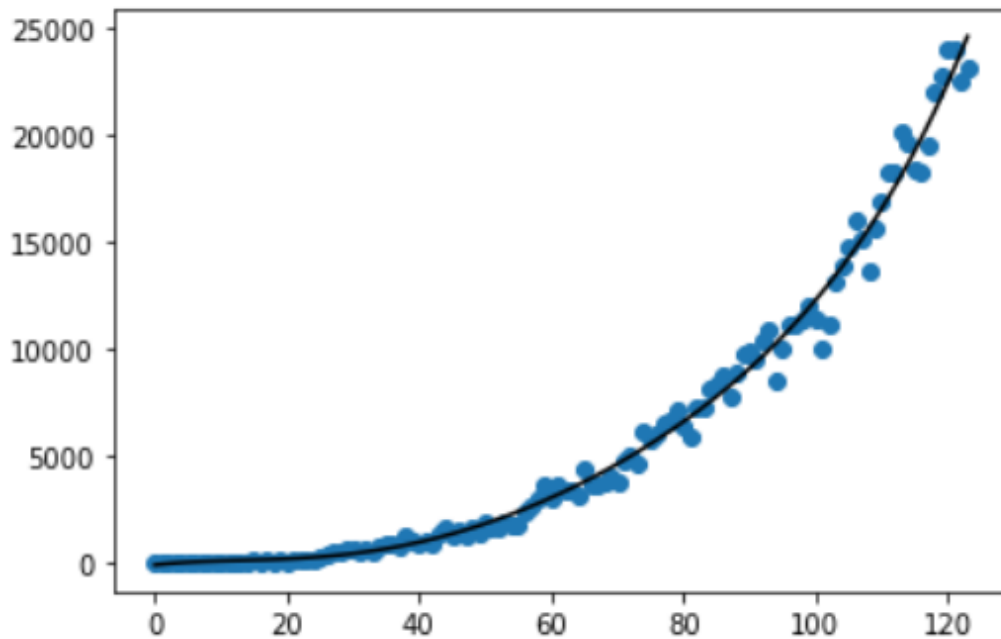
What we can see here is that number of affected males is double than the no. of females affected. This means that males have double the chance of contracting covid-19 than a female.

The data shows that age=30 is the highest infected group with a total of 1446 infected people. So, the people of age=30 have the maximum chance of getting Covid-19.

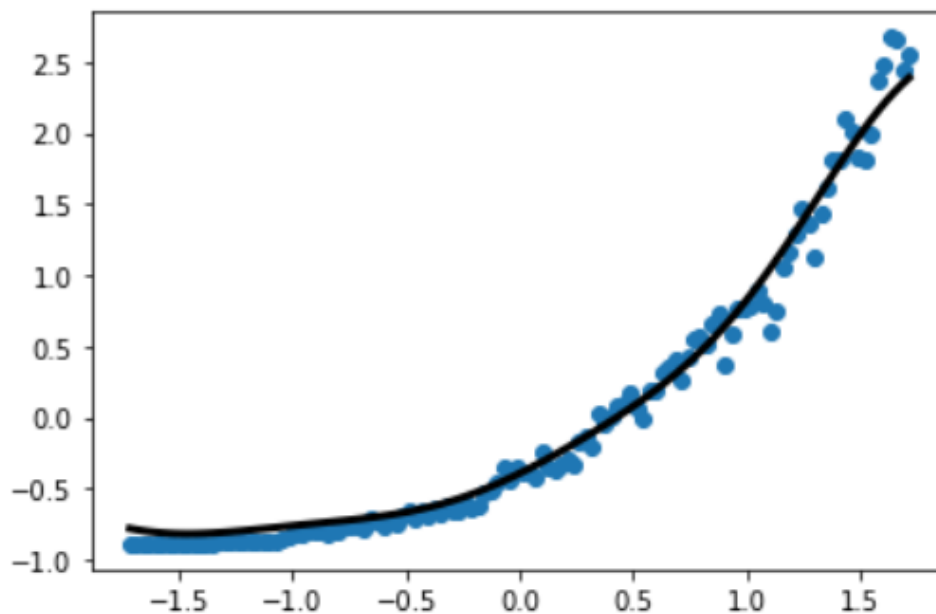
The diagram shows the prediction line using linear regression for cases that was finally presented.



The diagram shows the prediction line using Polynomial regression for cases that was finally presented.



The diagram shows the prediction line using Support Vector regression for cases that was finally presented.



Conclusion

The main objective of the Project is to study and analyze COVID-19 prevalence. The spread of the disease in India compared to other countries, state-wise trend of the epidemic to know how it is spreading and to analyze the healthcare sector of India and finally to predict the future of the epidemic in India.

This study report effectively examined the present global pattern of Covid-19 transmission. Anticipating the spread of Covid-19, also known as Novel Coronavirus, will aid in the implementation of appropriate control measures. The article also gave a complete research of the viral outbreak scenario in India, which will aid in taking the essential actions to control India's massive population. Three Machine Learning models were utilized for this, however in the future, Deep Learning models or a combination of two or more models might be used to anticipate the virus's spread.

For this research, conventional data was employed rather than real-time data since the data validity of conventional data is not heavily dependent on time and its reaction time requirements originate from the external world. Despite the fact that realtime data can be relaxed in a few instances Unlike traditional data, which must meet every situation, real-time data is roughly right, with performance measures determined as the number of transactions missing their deadlines per unit time.

Our investigation demonstrates that when the Linear Regression, Polynomial Regression Algorithm and Support Vector Machine Algorithm are compared with each other it is stated that linear regression gives an accuracy of 82 percent in which the data is not matching with the line at the starting and ending points, while

in polynomial regression we get to see an accuracy of 98 percent with second degree polynomial but when the degree is increased to 5 it gives 99 percent accuracy but as polynomial regression also uses linear for its implementation it is very difficult for it to make both increasing and decreasing curves for a single dataset, therefore at last we used SVM which is capable of producing good curves for both increasing and decreasing cases for a single given dataset and represent it on a single graph with an accuracy of 98 percent.

This project case study asks and answers the question, "Why has the data analysis for India been done state/UTs wise?"

Furthermore, this study begs the question, "Why is the death rate" despite being the second most populous country in the World, India has the lowest literacy rate in the world. "What is the world's most populated country?"

In addition, extra qualities might be incorporated in the research to improve accuracy during the procedure. Further research might include analyzing the India dataset to forecast the number of cases in the future and how the death rate fluctuates as the number of cases increases. I hope this report adds to the global response to this pandemic and provides some references for future research.

REFERENCES

- 1 Sun, Pengfei, Xiaosheng Lu, Chao Xu, Wenjuan Sun, and Bo Pan. "Understanding of COVID-19 based on current evidence." *Journal of medical virology* 92, no. 6 (2020): 548- 551.
- 2 World Health Organization. "Coronavirus disease 2019 (COVID-19): situation report, 72." (2020).
- 3 Bhatia, Surbhi, Poonam Chaudhary, and NilanjanDey. "Introduction to Opinion Mining." In *Opinion Mining in Information Retrieval*, pp. 1-22. Springer, Singapore, 2020.
- 4 Gulati, Hina. "Predictive analytics using data mining technique." In *2015 2nd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 713-716. IEEE, 2015.
- 5 Seber, George AF, and Alan J. Lee. *Linear regression analysis*. Vol. **329**. John Wiley & Sons, 2012.
- 6 Peng, Liangrong, Wuyue Yang, Dongyan Zhang, ChangjingZhuge, and Liu Hong. "Epidemic analysis of COVID-19 in China by dynamical modeling." *arXiv preprint arXiv:2002.06563* (2020).
- 7 Bouighoulouden, Amine, and IlhamKissani. "CROP YIELD PREDICTION USING K- MEANS CLUSTERING." (2020).
- 8 Gupta, Sonal, Gourav Singh Raghuwanshi, and Arnab Chanda. "Effect of weather on COVID-19 spread in the US: a prediction model for India in 2020." *Science of The Total Environment* (2020): 138860.

