

# **A Project/Dissertation Review Report**

on

## **COVID-19 DATA ANALYSIS**

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

# **BACHELOR OF ENGINEERING**



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

**Under The Supervision of**  
**Name of Supervisor: Dr. S.Srinivasan**  
**Designation: Professor**

Submitted By

**HARSHIT KAUSHIK**  
(18021011642/18SCSE1010411)

**MANISH TIWARI**  
(18021011910/18SCSE1010687)

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

**INDIA  
OCTOBER, 2021**

## **CANDIDATE'S DECLARATION**

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled “**Covid-19 Data Analysis**” 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 2 months, under the supervision of **Dr. S.Srinivasan** Designation - **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.

Harshit Kaushik 18SCSE1010411 Manish  
Tiwari 18SCSE1010687

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

Dr.S.Srinivasan  
Professor

## **CERTIFICATE**

The Final Thesis/Project/ Dissertation Viva-Voce examination of Harshit Kaushik 18SCSE1010411 and Manish Tiwari 18SCSE1010687 has been held on 05-12-2021 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: 05-12-2021  
Place: Greater Noida

## Submitted/Communicated/Accepted/Published Proof

Search all conversations

Active

1 of 650

IEEE IAS Global Conference on Emerging Technologies (GlobConET) : Submission (112) has been created.

External Inbox x

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

6:08 PM (4 minutes ago)

Hello,

The following submission has been created.

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

Paper ID: 112

Paper Title: Covid-19 Data analysis and visualization

Abstract:  
On March 11 2020 the World Health Organization proclaimed COVID 19 often known as Corona virus, to be a pandemic This infectious disease was identified in December 2019 in Wuhan, China, and has afflicted millions of individuals in the whole world Every country in the world is going through a worldwide economic crisis, thus it's more important than ever to estimate the frequency and occurrence of this pandemic throughout This would assist Indian doctors and government organizations in making critical judgments and taking proper actions to neutralize the disease and keep the country out of the economic downturn, This research seeks to visualize the number of cases in India and across the whole world using machine learning techniques and data analysis to map the increase in the frequency of those infected and discover growth trends The data came from the official COVID 19 website, which displayed verified sick cases from all Indian States, and Countries like China Korea etc The total number of confirmed cases from beginning March 2020 to 3 October 2021 will be used to determine how helpful present efforts have been, as well as the need to fight more to battle this virus The goal of this study is to anticipate the number of confirmed cases till date utilizing data visualization and data analysis techniques such as data analysis The main goal is to forecast the number of cases in the future month.

Created on: Fri, 17 Dec 2021 12:38:31 GMT

Authors:

- [harshit\\_kaushik.scsebtch@galgotiasuniversity.edu.in](mailto:harshit_kaushik.scsebtch@galgotiasuniversity.edu.in) (Primary)
- [manish\\_tiwari.scsebtch@galgotiasuniversity.edu.in](mailto:manish_tiwari.scsebtch@galgotiasuniversity.edu.in)

Secondary Subject Areas: Not Entered

Submission Files: Research Paper.pdf (439 Kb, Fri, 17 Dec 2021 12:36:17 GMT)

Submission Questions Response: Not Entered

Thanks,  
CMT team.

Download the CMT app to access submissions and reviews on the move and receive notifications:

<https://apps.apple.com/us/app/conference-management-toolkit/id1532488001>

<https://play.google.com/store/apps/details?id=com.microsoft.research.cmt>

To stop receiving conference emails, you can check the 'Do not send me conference email' box from your User Profile.

Microsoft respects your privacy. To learn more, please read our [Privacy Statement](#).

Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052

## 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**

The outbreak of COVID-19 in different parts of the world is a major concern for all the administrative units of respective countries. India is also facing this very tough task for controlling the virus outbreak and has managed its growth rate through some strict measures. India reported its first Covid-19 case on 30th Jan 2020 and the number of cases reported heavily escalated from March, 2020.

This project on **COVID -19 data analysis** is initially at a global level and then drills down to the scenario obtained in India. Data is gathered from multiple data sources-several authentic government websites. The need of the hour is to accurately forecast when the numbers will reach at its peak and then diminish. It will be of huge help to public welfare professionals to plan the preventive measures to be taken keeping the economic balance of the country as well. Variables A comparative analysis is also done to understand which model fits the best for our data. Data is considered till date. The project is giving an estimate of the day on which we can expect the number of active cases to reach its peak and also when the curve will start to flatten. This is unique feature of analysis in this project.

What we are trying to do in this project is that we are going to import an already given data in the form of a CSV and XLSX file and then visualise it constantly at every step using an Python package known as covid19.analytics which is a library collection for covid19 data function and it will show the deaths recovered latitude and longitude and various details of that place. We also use different libraries in this like pandas , matplotlib , plotly and many more

We will have time series Forecasting techniques and we will analyzing the summary reports of different countries in same time period which includes death rate, growth rate, recovered and a comparative analysis is also done to understand which model fits best for our data.

## **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

## **Introduction**

COVID-19, or more commonly known as the Novel Corona Virus is transmitted to the "Nidovirus" family, or "Nidovirales". COVID-19 is associated with a respiratory disorder that has been declared a global epidemic in the first quarter of 2020 by the World Health Organization. The most common signs or symptoms of COVID-19 are fever, fatigue, dry cough, pain and soreness, nasal congestion, runny nose or sore throat. COVID-19 is a "contagious" disease and can be passed through droplets from the nose or mouth when an infected person coughs or breathes and this is a major reason to keep a distance of 1m (3 feet) from the infected person.

According to the latest data, there are currently more than 48.8 million people infected with the Novel Corona Virus worldwide and approximately 145M people are reported from different parts of the world. On January 30, 2020, India reported its first case of coronavirus in Kerala when a student returned from Wuhan (epicenter of coronavirus) and until then the number of cases had increased dramatically. In recent times there is vaccine available especially for the treatment of COVID-19 and it is currently under investigation. This paper analyses the current practice of COVID-19.

## **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 outspread 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 available in the literature, there are some studies that focus on trend analysis and forecasting for the Indian region. The study on the Indian region presents long-term and short-term trends respectively. These studies use time series data from the John Hopkins University database and present forecasts using the ARIMA model, exponential smoothing methods, the SEIR model, and the regression model. In addition, studies in the Indian region from the past are more focused on presenting time series analysis based on aggregate data for the Indian region.

Similarly, there are other mathematical models that were developed to analyze COVID-19 outbreak trends in India. A model was presented to study the effect of social distinction on age and gender of patients in India. It compared the demographics of the country between India, Italy and China and suggested the weakest age categories and gender groups among all countries. The study also predicted an increase in infected cases with different lockdown periods in India. Similarly, a network structure approach was used by one of the studies to see if a specific node cluster was formed. But only travel data nodes were considered by the authors to examine which major areas are affecting Indian travellers returning to India. Also, the study presented the SIR model to see the rate of spread of corona virus among patients in India. Analysis on testing laboratories and infrastructure was also presented by earlier authors.

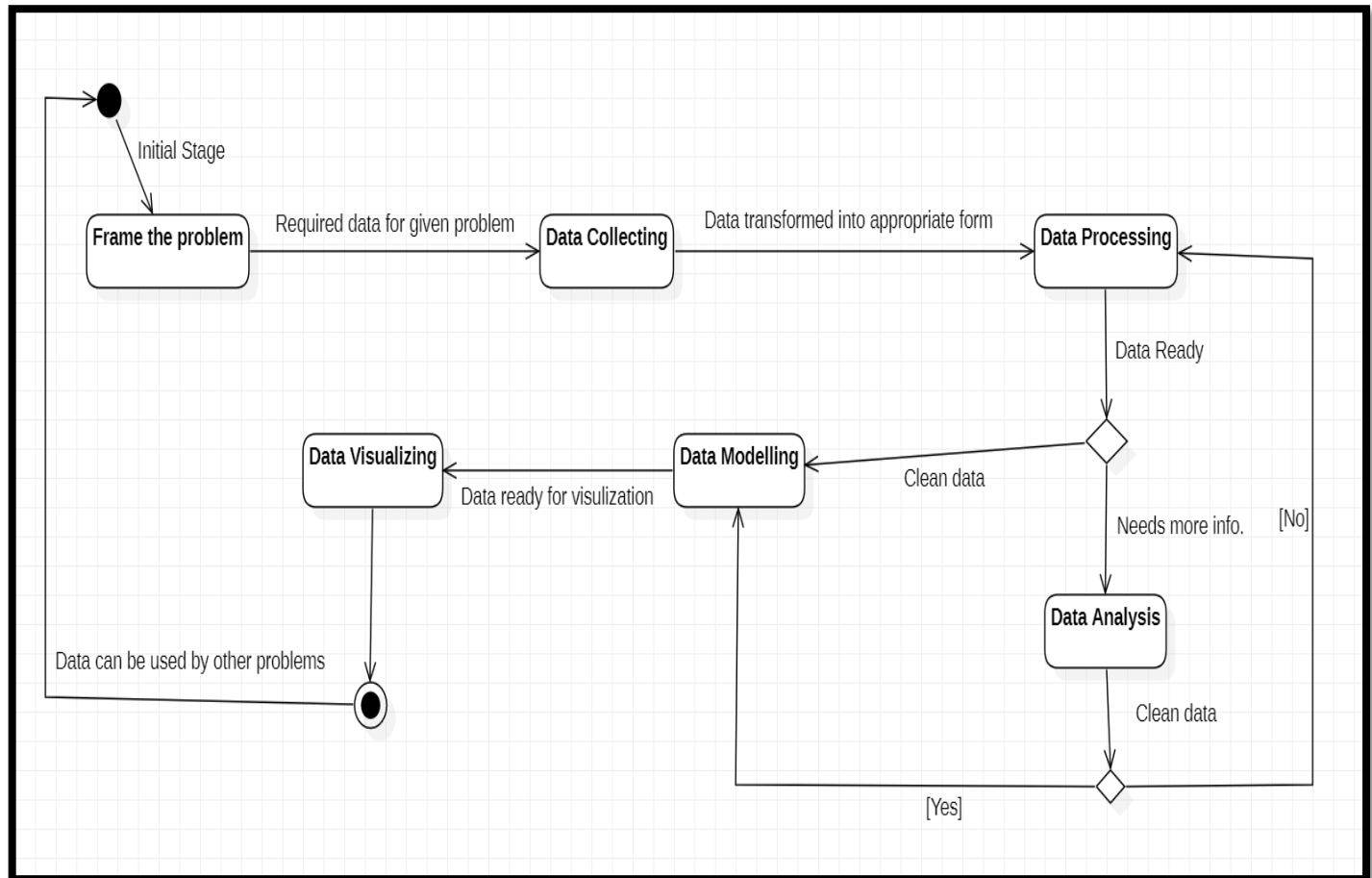
The work of medical doctors and frontline health workers was also presented by some studies. It was found that in India, the role of health workers was less emphasized as the stages of the spread of the corona virus were still in two or more stages, such as community broadcasting compared to other countries such as Italy, Spain and the United States .

Apart from India, some models are also available for other countries mainly for China, Italy and USA as the number of infected patients was high. The study worked on various mathematical models to determine the spread of the disease, predicted the number of infected patients, commented on each country's preparedness to deal with the spread of COVID-19, and the pattern of the flattened curve under different conditions Searched A lot of research for the world stage is still in its first phase and is yet to be reviewed.

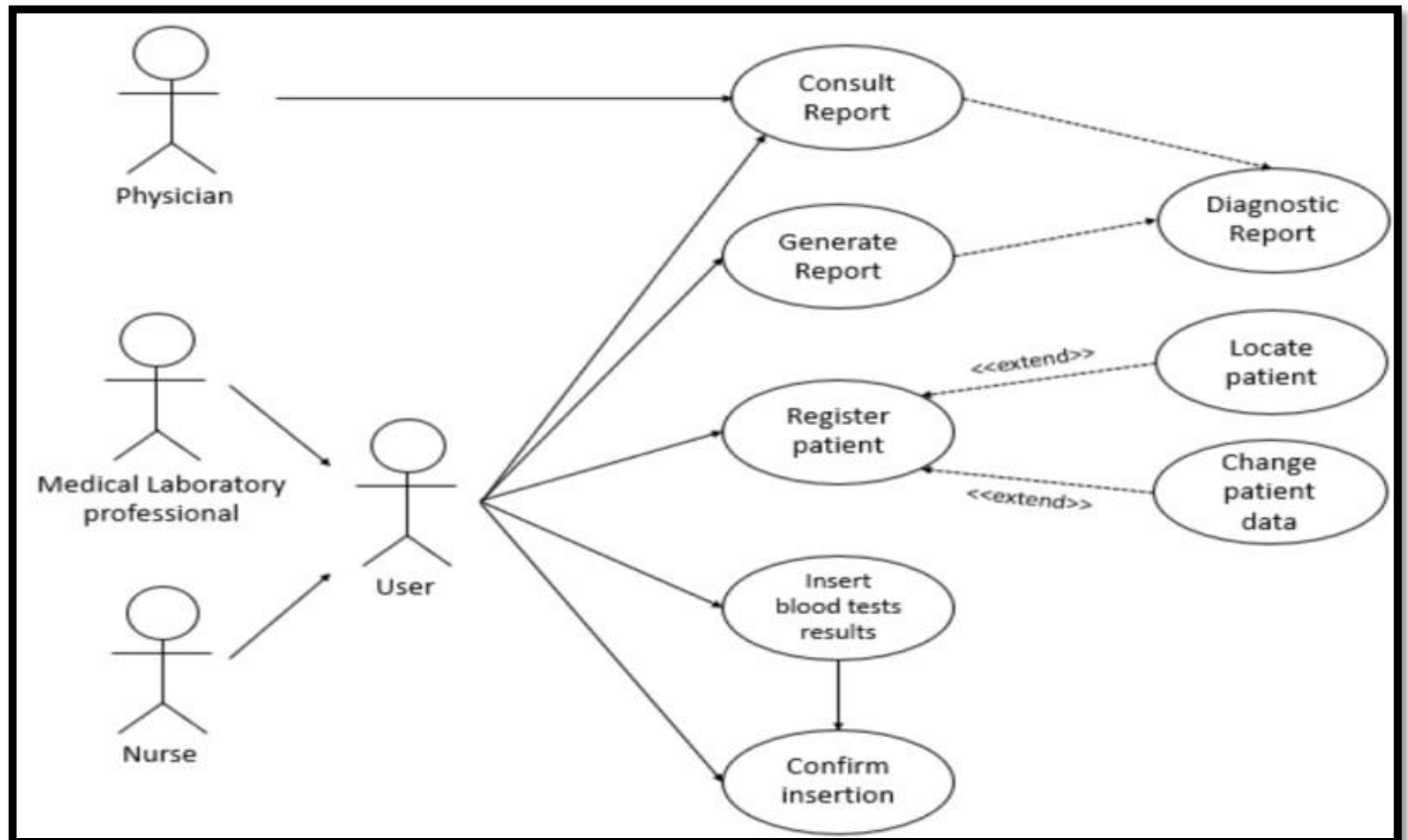
In relation to the research activities conducted in the Indian region, the study remains to work on the impact of various policies working towards the control of the corona virus. Therefore, this study attempts to work on a broader level for the analysis of COVID19 spread in India.

# Project Design

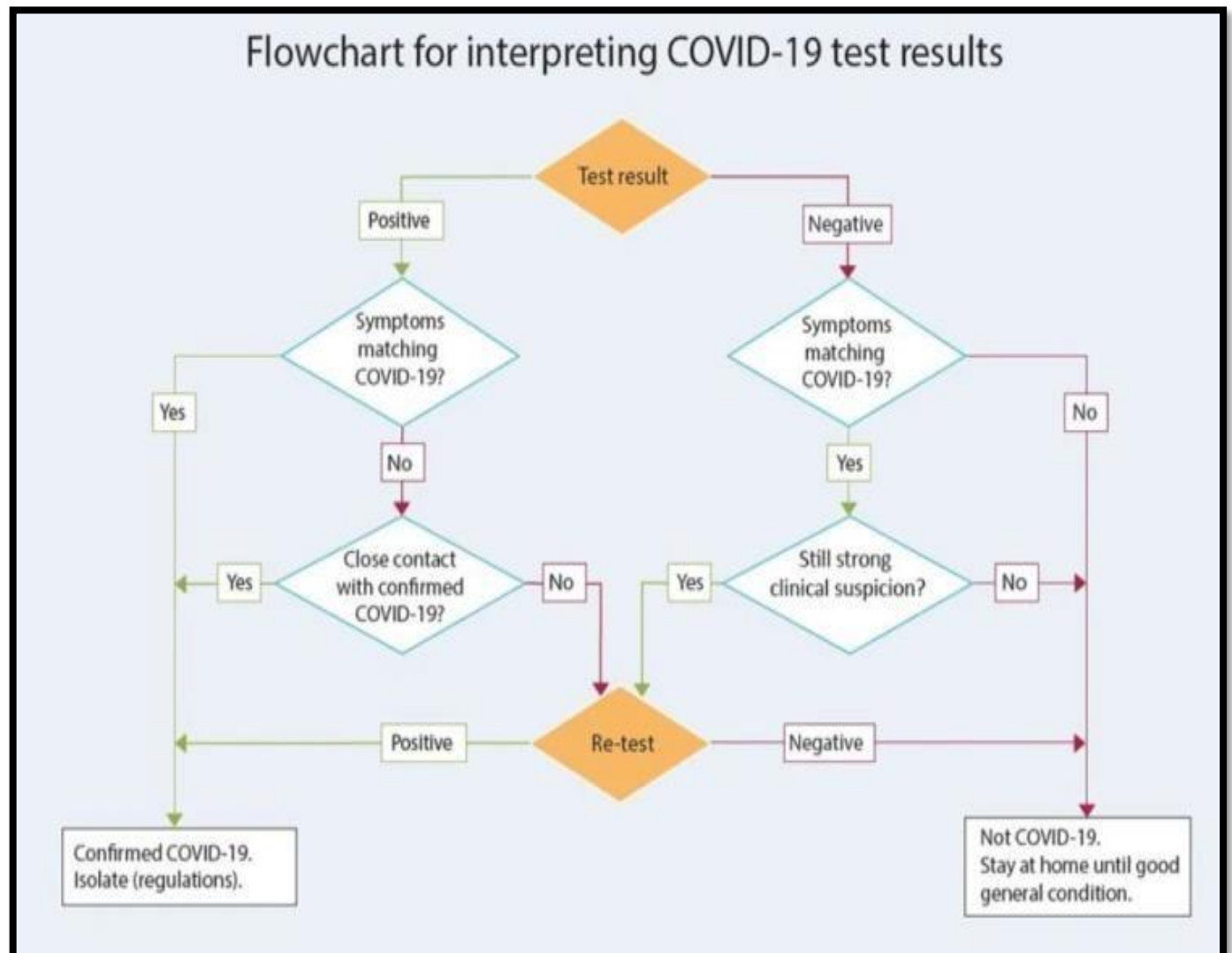
## UML Diagram



## 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:**

- Anaconda Navigator
- Jupyter Notebook
- Libraries/Packages:
  - Pandas
  - matplotlib.pyplot
  - matplotlib
  - plotly
  - cufflinks
  - folium

## **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:

$$y = B0 + B1*x$$

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.

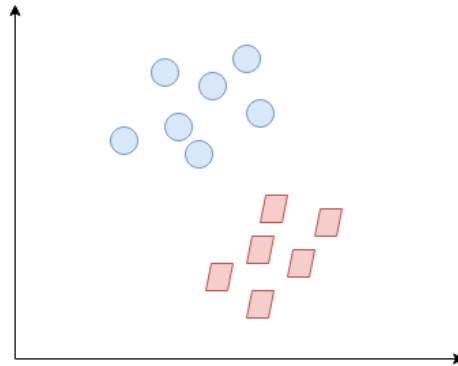
The training dataset for polynomial regression is non-linear in character.

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

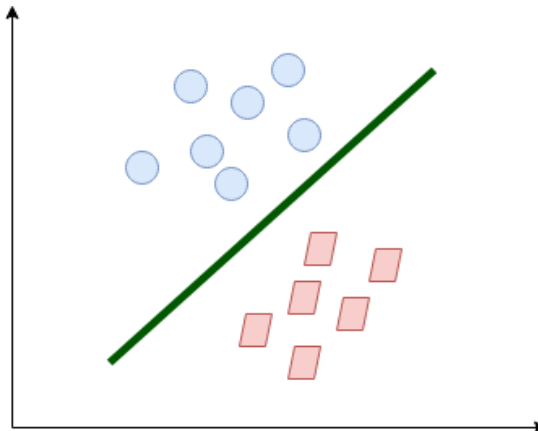
### iii) Support Vector Regression

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

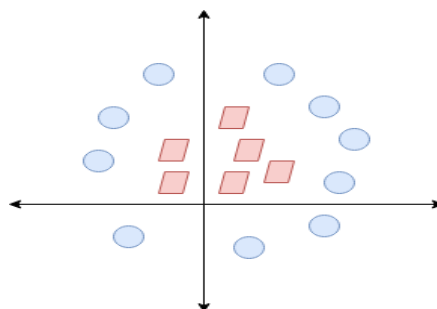
So, what is a Support Vector Machine (SVM) and how does it work? Let's start with a basic understanding of SVM. Assume we have a plot with two label classes, as illustrated in the diagram:



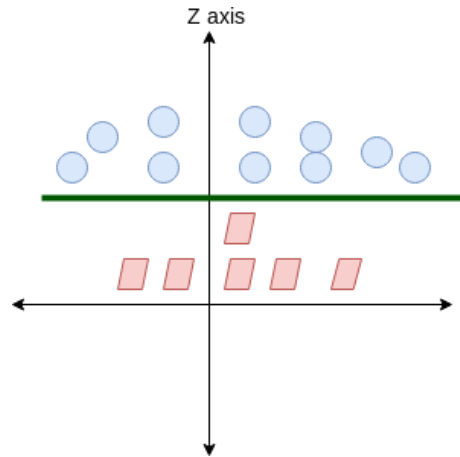
Can you decide where the dividing line will be drawn? You could have thought of this:



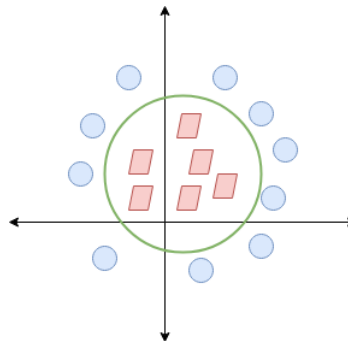
The line effectively divides the classes. Simple class separation is essentially what SVM accomplishes. Now, here's what the data looked like:



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.

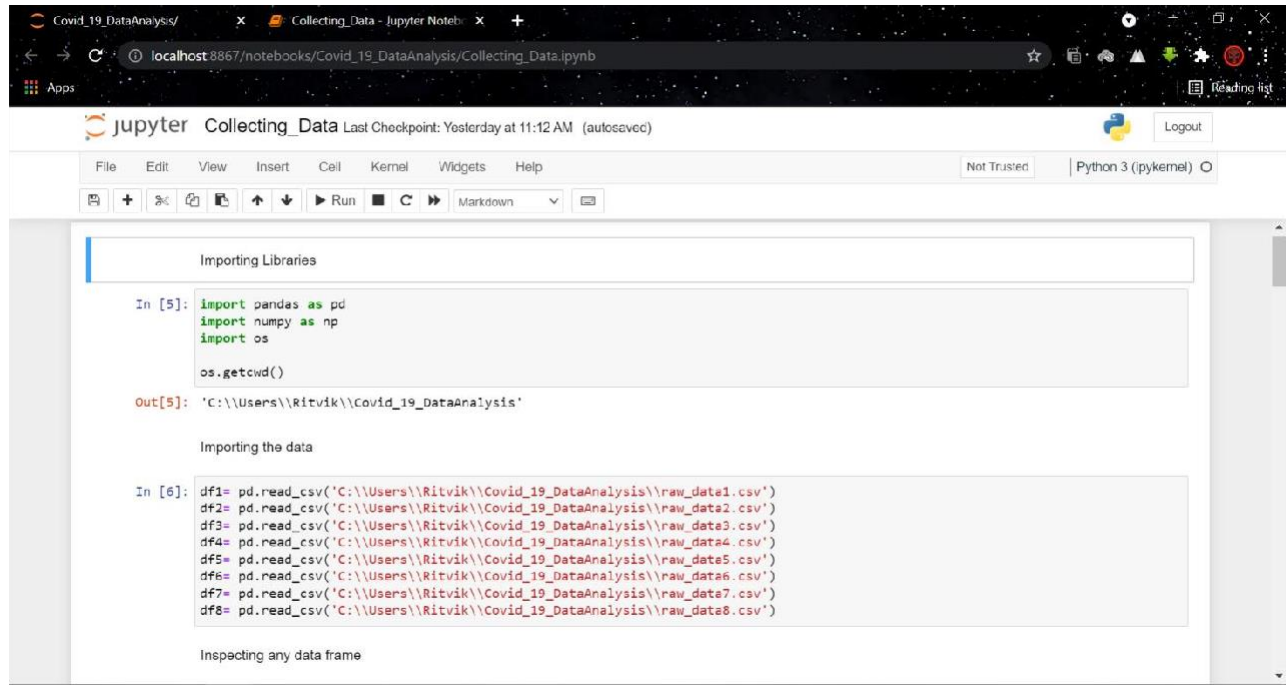
Support Vector Regression (SVR) is a regression technique that uses the same principles as SVM. Let's take a few moments to grasp the concept of SVR. On the basis of a training sample, the aim of regression is to identify a function that approximates mapping from an input domain to real numbers. So let's 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.



# Implementation and Testing



```
Importing Libraries

In [5]: import pandas as pd
import numpy as np
import os

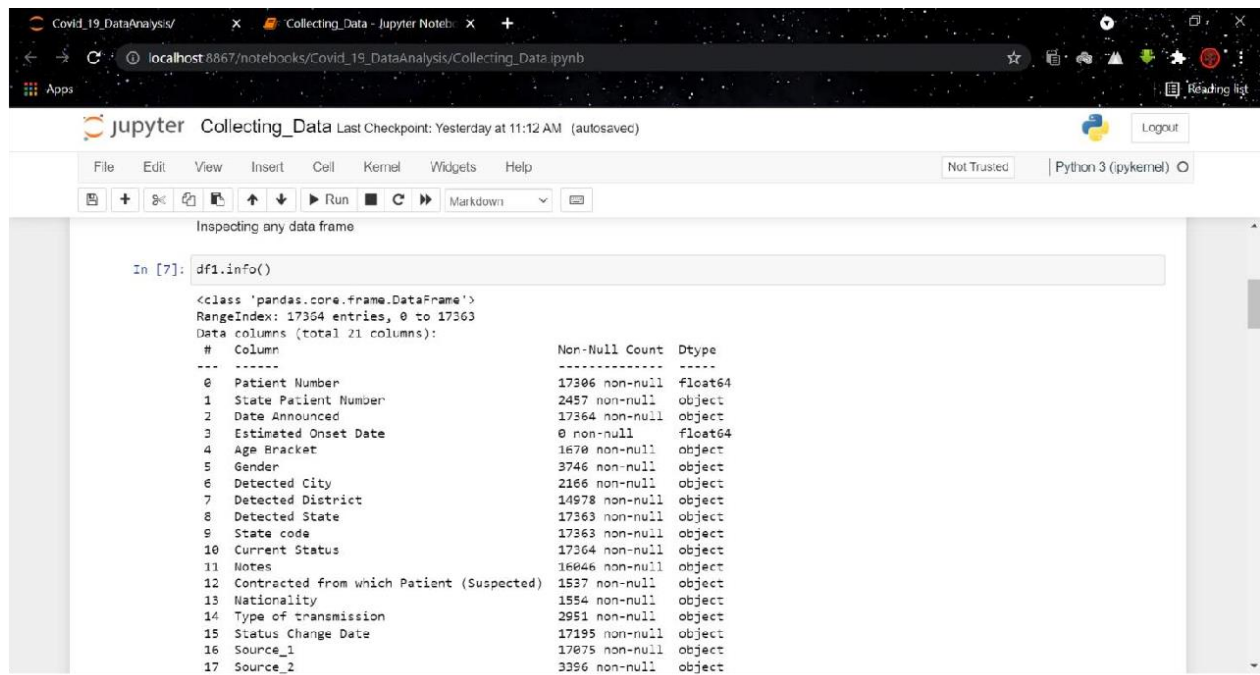
os.getcwd()

Out[5]: 'C:\\Users\\Ritvik\\Covid_19_DataAnalysis'

Importing the data

In [6]: df1= pd.read_csv('C:\\Users\\Ritvik\\Covid_19_DataAnalysis\\raw_data1.csv')
df2= pd.read_csv('C:\\Users\\Ritvik\\Covid_19_DataAnalysis\\raw_data2.csv')
df3= pd.read_csv('C:\\Users\\Ritvik\\Covid_19_DataAnalysis\\raw_data3.csv')
df4= pd.read_csv('C:\\Users\\Ritvik\\Covid_19_DataAnalysis\\raw_data4.csv')
df5= pd.read_csv('C:\\Users\\Ritvik\\Covid_19_DataAnalysis\\raw_data5.csv')
df6= pd.read_csv('C:\\Users\\Ritvik\\Covid_19_DataAnalysis\\raw_data6.csv')
df7= pd.read_csv('C:\\Users\\Ritvik\\Covid_19_DataAnalysis\\raw_data7.csv')
df8= pd.read_csv('C:\\Users\\Ritvik\\Covid_19_DataAnalysis\\raw_data8.csv')

Inspecting any data frame
```

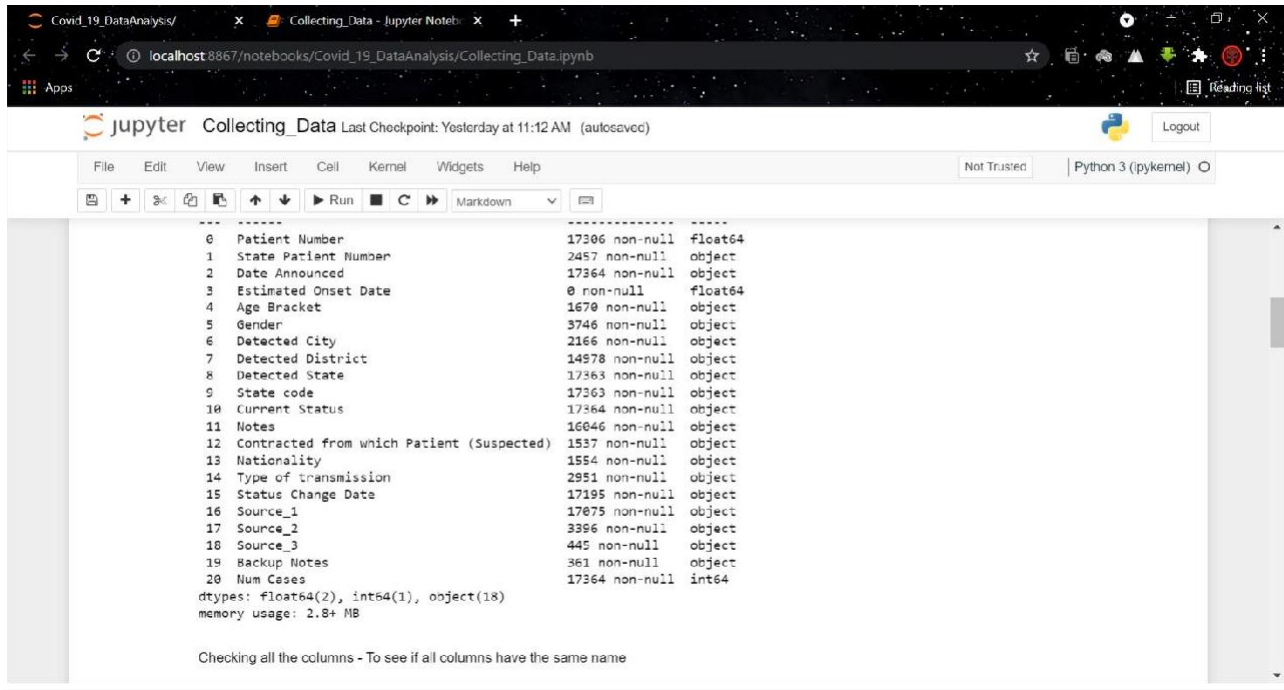


```
Inspecting any data frame

In [7]: df1.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 17364 entries, 0 to 17363
Data columns (total 21 columns):
#   Column                                     Non-Null Count  Dtype
---  -
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                                     16646 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                                 17675 non-null  object
17  Source_2                                 3396 non-null  object
```

# Covid-19 Data Analysis

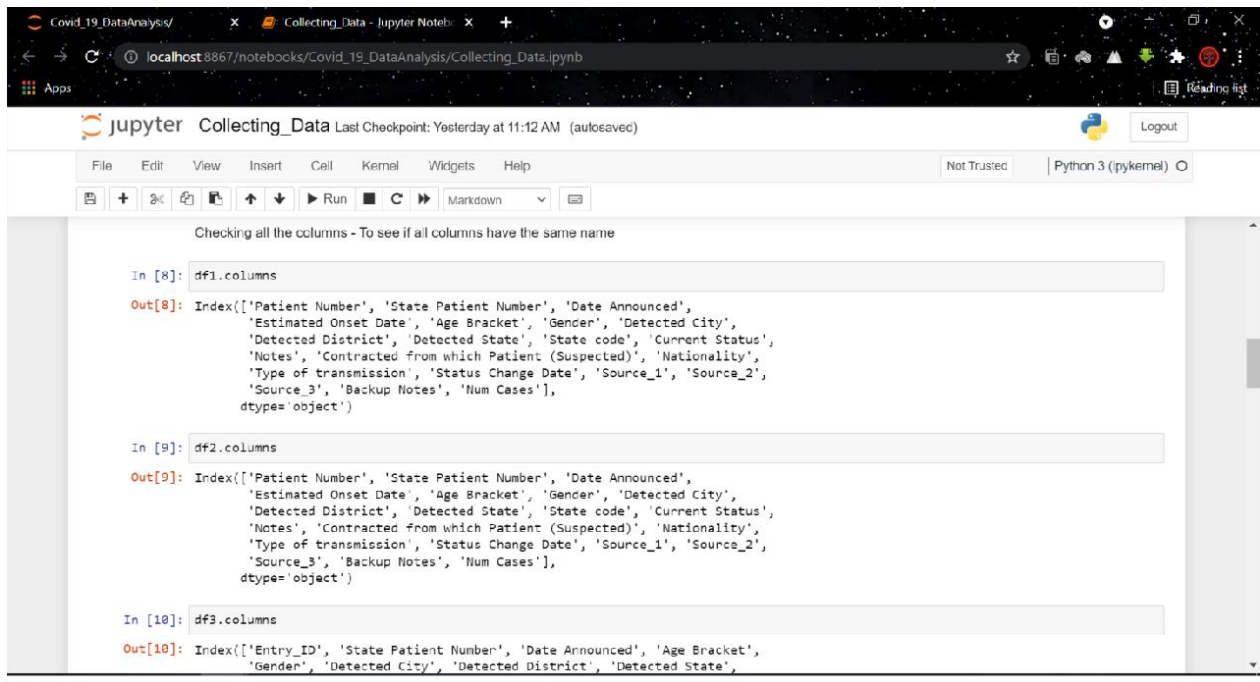


The screenshot shows a Jupyter Notebook interface with a table of data statistics. The table lists 20 columns with their respective counts, null values, and data types. Below the table, it shows the dtypes for each column and the total memory usage.

Column Index	Column Name	Count	Null Value	Data Type
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



The screenshot shows a Jupyter Notebook with three code cells. Each cell checks the columns of a different DataFrame (df1, df2, df3) to see if they match the columns of the first DataFrame (df1). The output shows the column names and their data types for each DataFrame.

```
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 interface with the following content:

```
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
```

Merging all data frames

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

```
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

Collecting\_Data Last Checkpoint: Yesterday at 11:12 AM (autosaved)

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

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

Collecting\_Data Last Checkpoint: Yesterday at 11:12 AM (autosaved)

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

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 (Mayapuri/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 12 columns

# 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	-5.0	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
...	...	...	...	...	...	...	...	...	...	...	...	...
145344	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 (Mayapuri Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...	...	...	...	...	...	...	...	...	...	...	...
145344	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Diu	Hospitalized	7	7	2020
145345	-8.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145346	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145347	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	7	7	2020
145348	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         60613 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
```

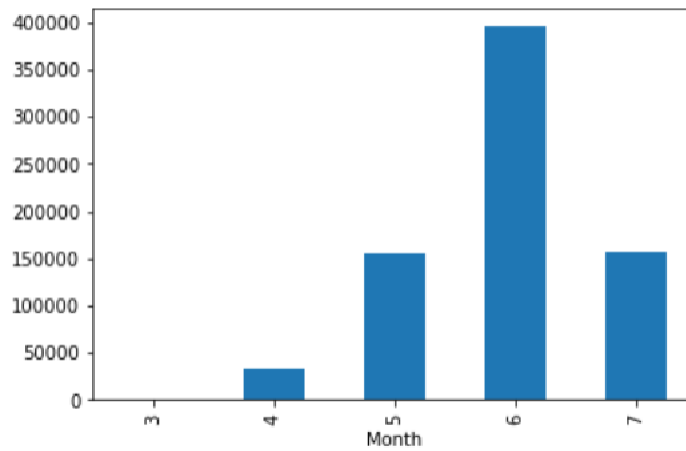
```
In [7]: M = data[data['Current Status']=='Hospitalized'].groupby('Month')['Num Cases'].sum()
```

```
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()
```



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)

### Total Male/Female affected by coronavirus

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

```
Out[9]: Gender  
F          21294.0  
M         42795.0  
H              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
```

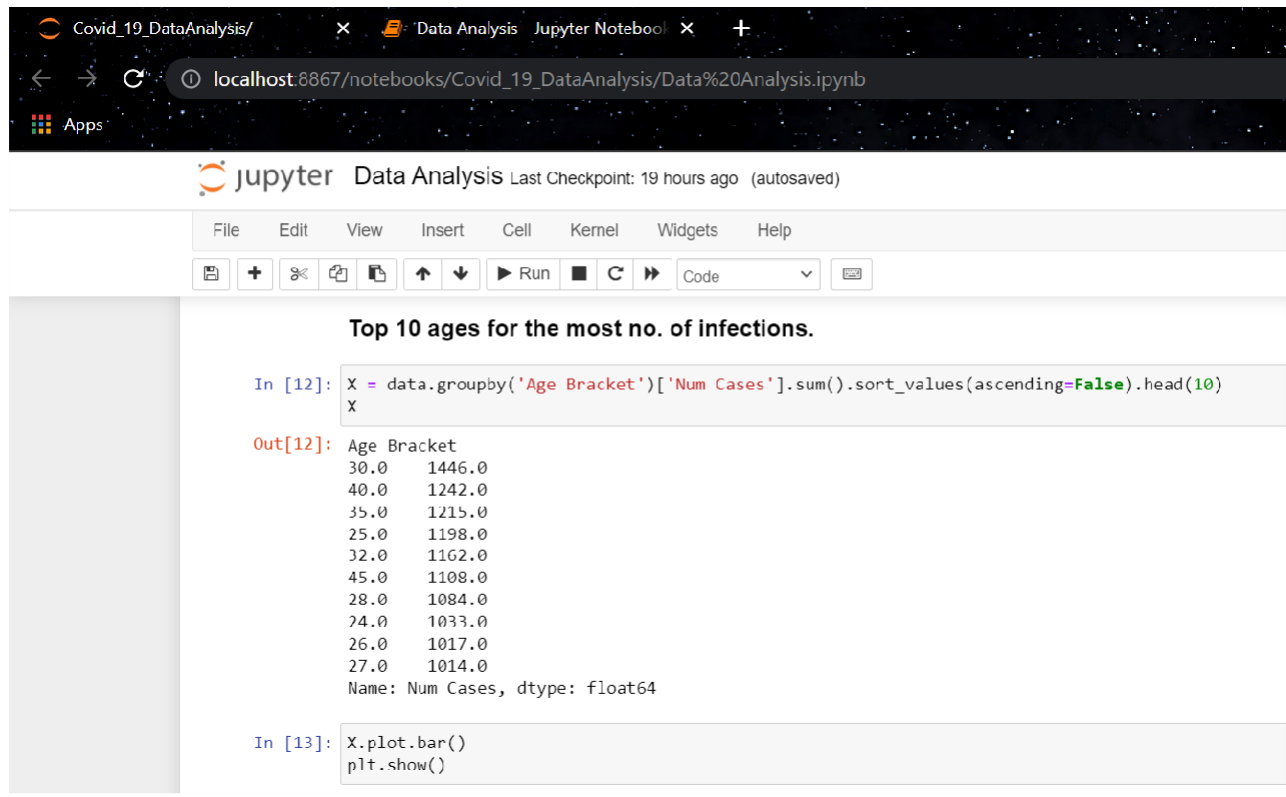
Now we will sort the above data in descending order to find the age with the highest no. of infected people.

```
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
```

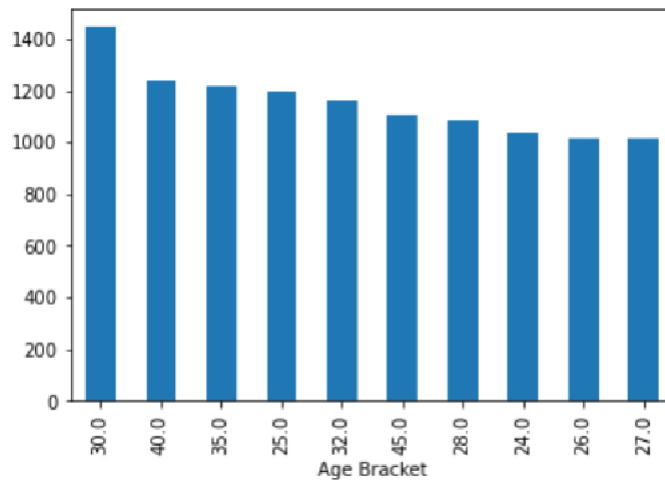
The above data shows that the age=30 is the highest infected group with a total of 1446 infected people. So, the people of age=30 have the maximum chance of getting Covid-19.

# Covid-19 Data Analysis



The screenshot shows a Jupyter Notebook window titled "Data Analysis" with a last checkpoint of 19 hours ago. The notebook contains two code cells. The first cell, labeled "In [12]:", executes the following code: `X = data.groupby('Age Bracket')['Num Cases'].sum().sort_values(ascending=False).head(10)`. The output, labeled "Out[12]:", displays a table of the top 10 age brackets and their corresponding number of cases. The second cell, labeled "In [13]:", executes `X.plot.bar()` and `plt.show()` to generate a bar chart.

```
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()
```



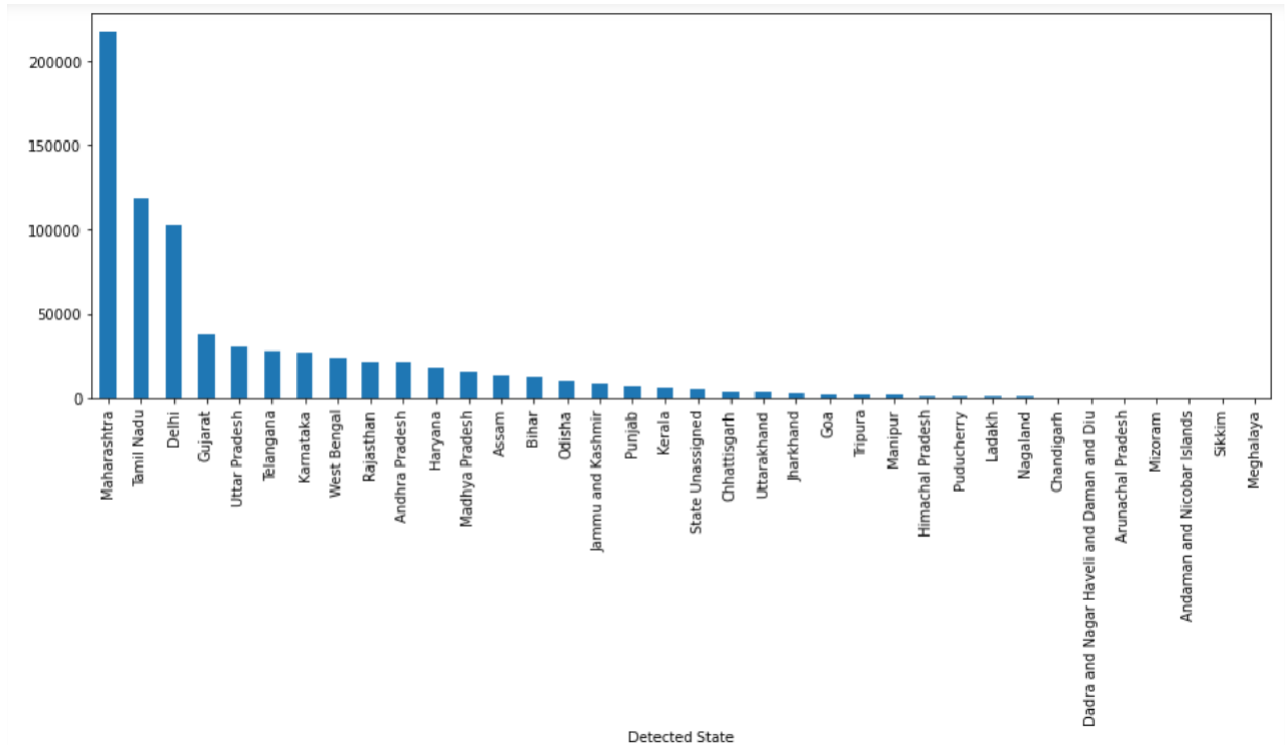
# Covid-19 Data Analysis

**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	Num Cases
Maharashtra	217107.0
Tamil Nadu	118597.0
Delhi	102827.0
Gujarat	47631.0
Uttar Pradesh	20050.0
Telangana	27010.0
Karnataka	26743.0
West Bengal	23831.0
Rajasthan	21400.0
Andhra Pradesh	21195.0
Haryana	17987.0
Madhya Pradesh	15025.0
Assam	14337.0
Bihar	12524.0
Odisha	10096.0
Jammu and Kashmir	8930.0
Punjab	6747.0
Kerala	5834.0
State Unassigned	5034.0
Chhattisgarh	3107.0
Uttarakhand	3229.0
Jharkhand	3018.0
Goa	1903.0
Tripura	1716.0
Manipur	1429.0



# 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)

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	4.0
	10	4.0
...	...	...
3		22718.0
4		24018.0
7	5	23942.0
	6	22500.0
	7	23147.0

124 rows x 1 columns

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)

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()
```

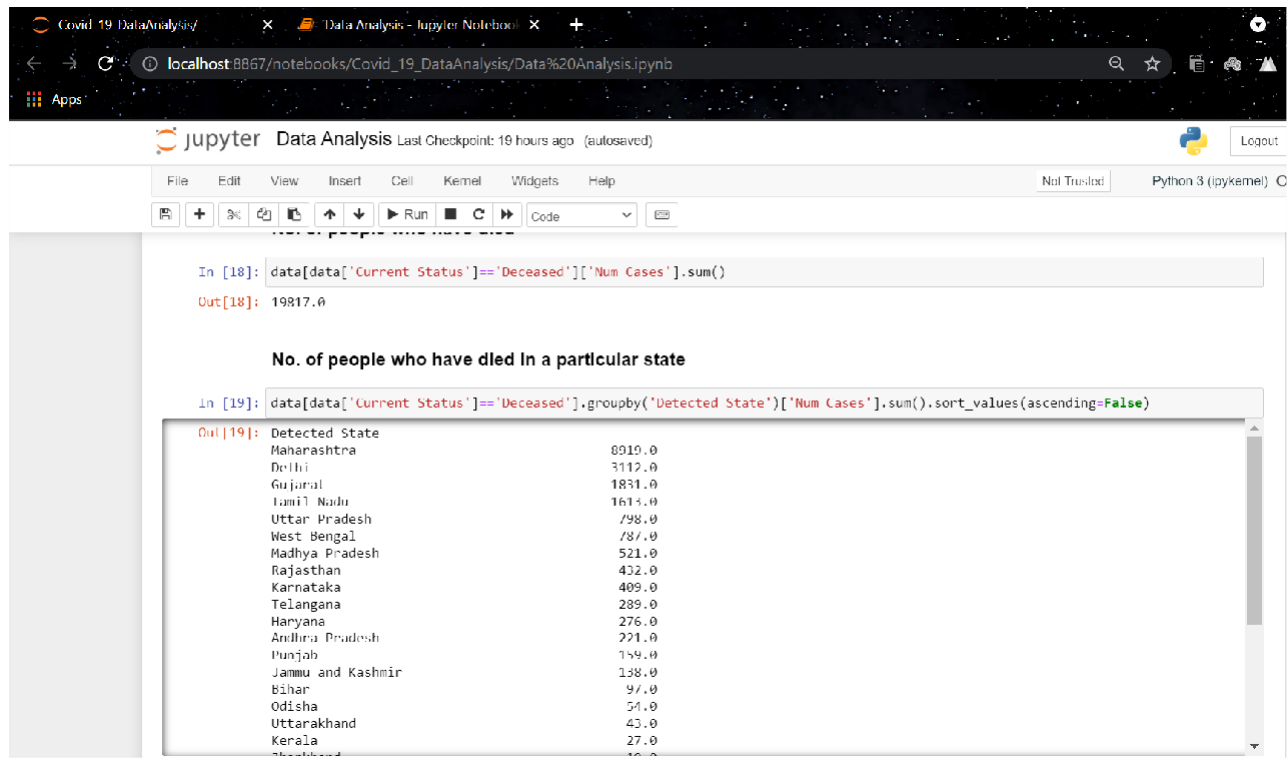
(Num Cases, 3)

(Num Cases, 4)

(Num Cases, 5)

(Num Cases, 6)

# Covid-19 Data Analysis

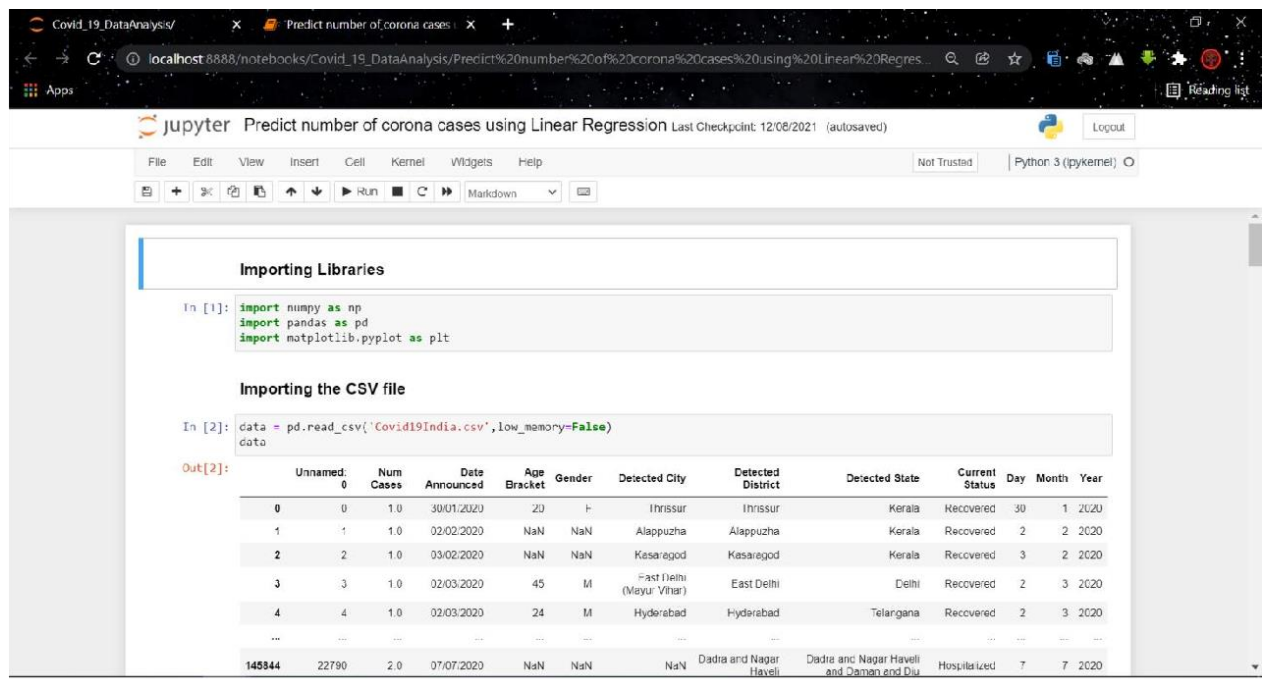


```
In [18]: data[data['Current Status']=='Deceased']['Num Cases'].sum()
Out[18]: 19817.0
```

**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	1615.0
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	271.0
Punjab	154.0
Jammu and Kashmir	138.0
Bihar	97.0
Odisha	51.0
Uttarakhand	45.0
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	Data 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 (Mayapuri Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...	...	...	...	...	...	...	...	...	...	...	...	...
148844	22790	2.0	07/07/2020	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	Thirissur	Thirassur	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 (Mayapuri Vihar)	East Delhi	Delhi	Recovered	2	3	2020
4	1.0	02/03/2020	24	M	Hyderabad	Hyderabad	Telangana	Recovered	2	3	2020
...	...	...	...	...	...	...	...	...	...	...	...
145344	2.0	07/07/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Daman and Diu	Hospitalized	7	7	2020
145345	-9.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145346	-6.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	7	7	2020
145347	-1.0	07/07/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	7	7	2020
145348	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	22718.0	22718.0
4	24018.0	24018.0
7	23942.0	23942.0
6	22500.0	22500.0
7	23147.0	23147.0

124 rows x 1 columns

# Covid-19 Data Analysis

Covid\_19\_DataAnalysis/ Predict number of corona cases using Linear Regression Last Checkpoint: 12/08/2021 (autosaved)

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

7 23147.0

124 rows x 1 columns

**Generating the value of X-axis variable (no. of days)**

```
In [5]: len(Day)
```

Out[5]: 124

**Generating an array equal to the length of Day**

```
In [6]: x = np.arange(len(Day))
```

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])

Covid\_19\_DataAnalysis/ Predict number of corona cases using Linear Regression Last Checkpoint: 12/08/2021 (autosaved)

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

```
In [7]: y= Day.values
```

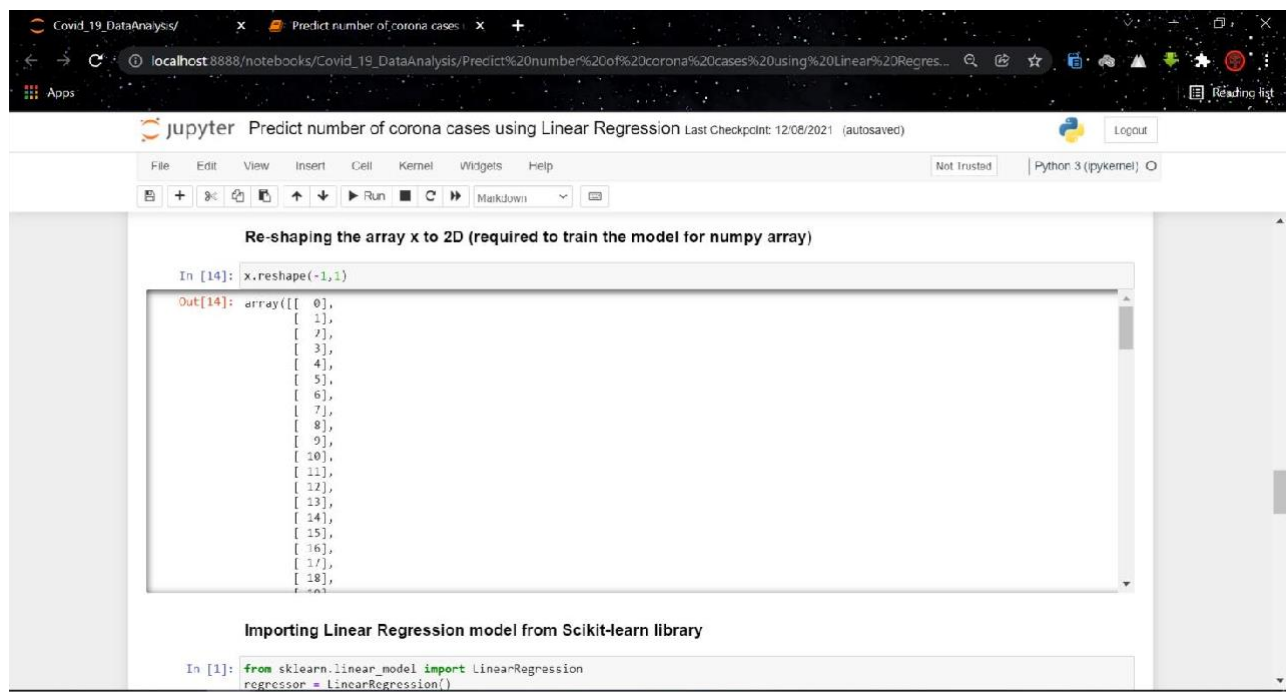
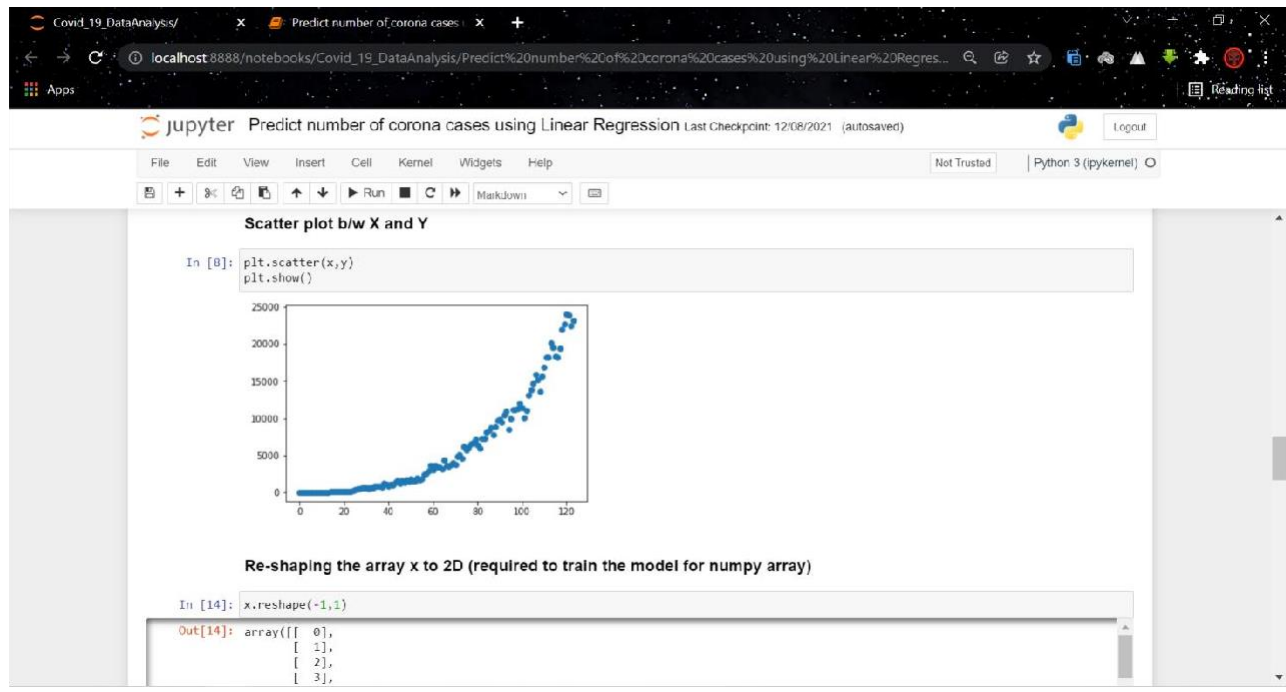
Out[7]: array([[5.0000e-00], [1.0000e-00], [2.0000e-00], [4.0000e-00], [4.0000e-00], [8.0000e-00], [4.0000e-00], [6.0000e-00], [1.1000e-01], [8.0000e-00], [1.2000e-01], [1.4000e-01], [2.2000e-01], [2.1000e-01], [5.2000e-01], [6.7000e-01], [5.9000e-01], [8.2000e-01], [6.3000e-01], [7.0000e-01]])

**Scatter plot b/w X and Y**

```
In [8]: plt.scatter(x,y)
```

plt.show()

# Covid-19 Data Analysis





# Covid-19 Data Analysis

Covid\_19\_DataAnalysis/ Predict number of corona cases using Linear Regression Last Checkpoint: 12/08/2021 (autosaved)

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

Now we will plot the best fit line over the the earlier plotted scatter plot

```
In [29]: Yp= regressor.predict(x)
        Yp
```

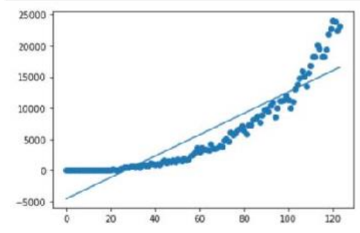
```
Out[29]: array([[ -4553.47612903],
                [-4381.97114083],
                [-4210.46615264],
                [-4038.96116444],
                [-3867.45617624],
                [-3695.95118004],
                [-3524.44619984],
                [-3352.94121164],
                [-3181.43622345],
                [-3009.93123525],
                [-2838.42624705],
                [-2666.92125885],
                [-2495.41627065],
                [-2323.91128245],
                [-2152.40629426],
                [-1980.90130606],
                [-1809.39631786],
                [-1637.89132966],
                [-1466.38634146],
                [-1294.88135326]])
```

```
In [31]: plt.scatter(x,y)
        plt.plot(x,Yp)
        plt.show()
```

Covid\_19\_DataAnalysis/ Predict number of corona cases using Linear Regression Last Checkpoint: 12/08/2021 (autosaved)

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

```
In [31]: plt.scatter(x,y)
        plt.plot(x,Yp)
        plt.show()
```



```
In [33]: regressor.score(x,y)*100
```

```
Out[33]: 82.93128436057233
```

# 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 (Mayapuri 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
06	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	73942.0
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

This screenshot shows the first two code cells of a Jupyter Notebook. The first cell creates an array of days from 0 to 123. The second cell shows the values of a 'Day' object for each day.

```
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
Out[121]: [ 4.740e-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],
 [ 1.8199e-04],
 [ 1.8256e-04],
 [ 1.9429e-04],
```

This screenshot shows the next three code cells of the Jupyter Notebook. The third cell reshapes the array 'x' into a 2D array. The fourth cell imports PolynomialFeatures and fits a polynomial transformation of degree 5. The fifth cell creates a DataFrame from the transformed array.

```
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],
 [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	200533021.0	2.389354e+10
120	1.0	120.0	14400.0	1728000.0	207360000.0	2.483320e+10
121	1.0	121.0	14641.0	1771561.0	214358881.0	2.583742e+10
122	1.0	122.0	14884.0	1815848.0	221533456.0	2.702708e+10
123	1.0	123.0	15129.0	1860867.0	228866641.0	2.815300e+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, -2.33294349e+03,  
                7.91412887e-02, -7.69939389e-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,  
                 4.33323049e+01,  
                 5.52546596e+01,  
                 6.57480494e+01,  
                 7.51433243e+01,  
                 8.37561074e+01,  
                 9.18871302e+01],
```

# Covid-19 Data Analysis

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

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

```
In [151]: regressor.score(X,y)*100
Out[151]: 99.04976030995705
```

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

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

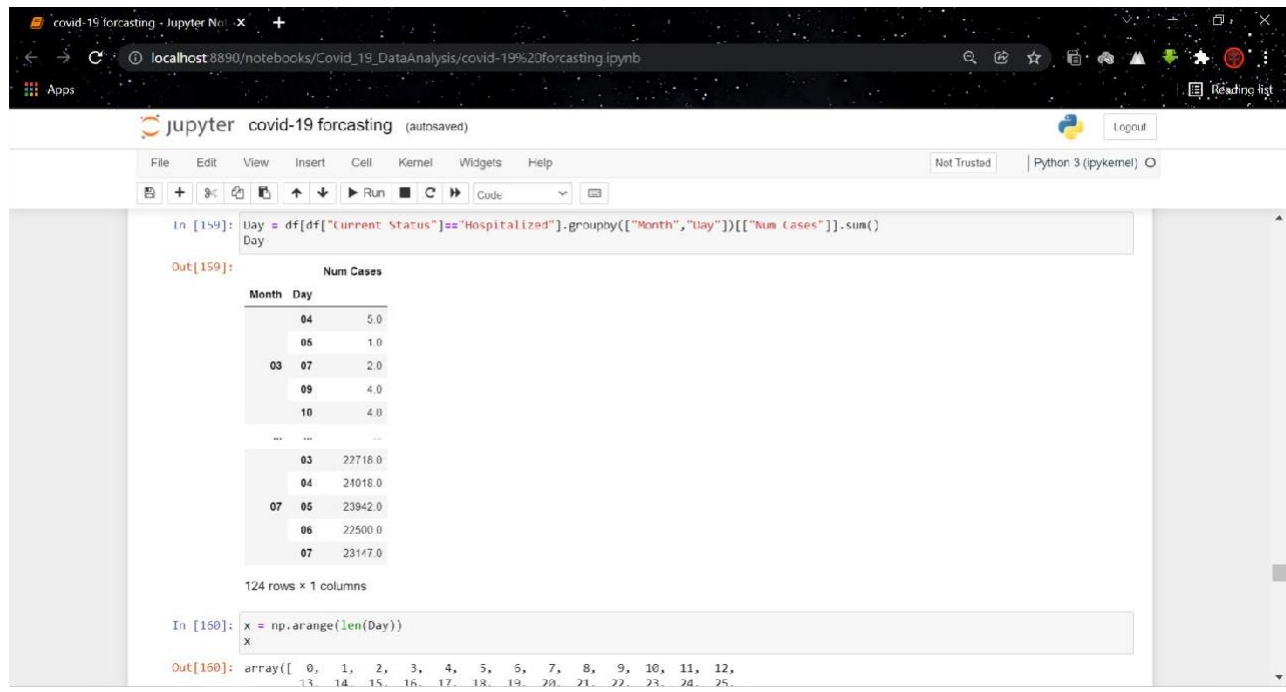
```
In [156]: regressor.predict(poly.transform([[125]]))
Out[156]: array([[26139.59234236]])
```

```
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	46	M	East Delhi (Mayapuri 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	01/01/2020	NaN	NaN	NaN	Dadra and Nagar Haveli	Dadra and Nagar Haveli and Diaman and Diu	Hospitalized	01	01	2020
22791	-0.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22792	-0.0	07/07/2020	NaN	NaN	NaN	NaN	Sikkim	Recovered	07	07	2020
22793	-1.0	01/01/2020	NaN	NaN	NaN	Kozhikode	Kerala	Recovered	01	01	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 [149]: day = df[df["Current Status"]=="Hospitalized"].groupby(["Month", "Day"])["Num Cases"].sum()
Day
```

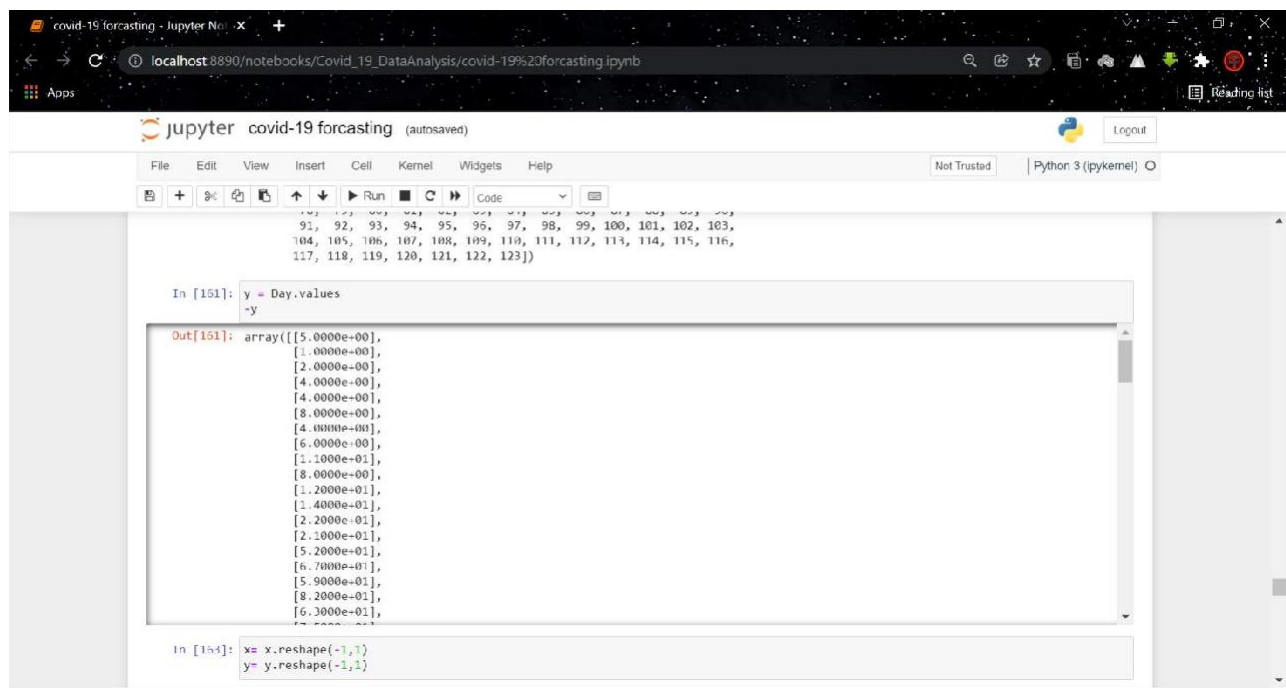
Out[149]:

Month	Day	Num Cases
04	05	5.0
05	06	1.0
03	07	2.0
09	09	4.0
10	10	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 [150]: x = np.arange(len(Day))
x
```

Out[150]: 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, ...])



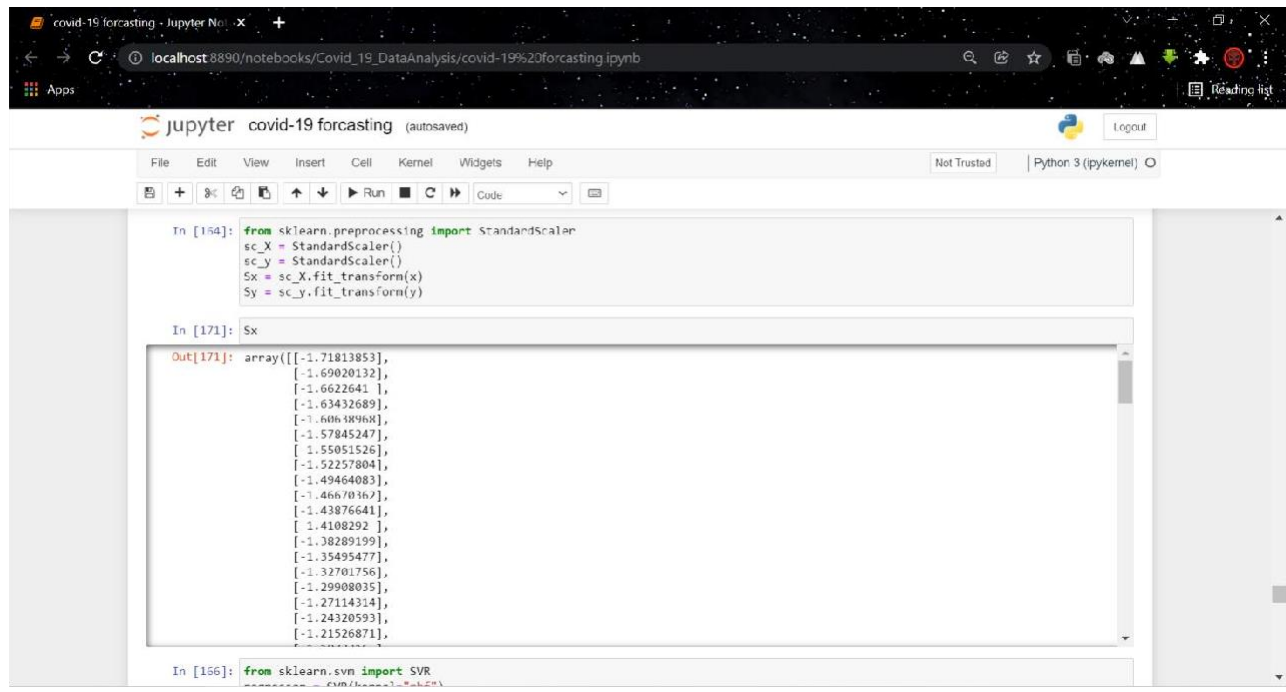
The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [151]: y = Day.values
-y
```

Out[151]: 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 [154]: x = x.reshape(-1,1)
y = y.reshape(-1,1)
```

# Covid-19 Data Analysis

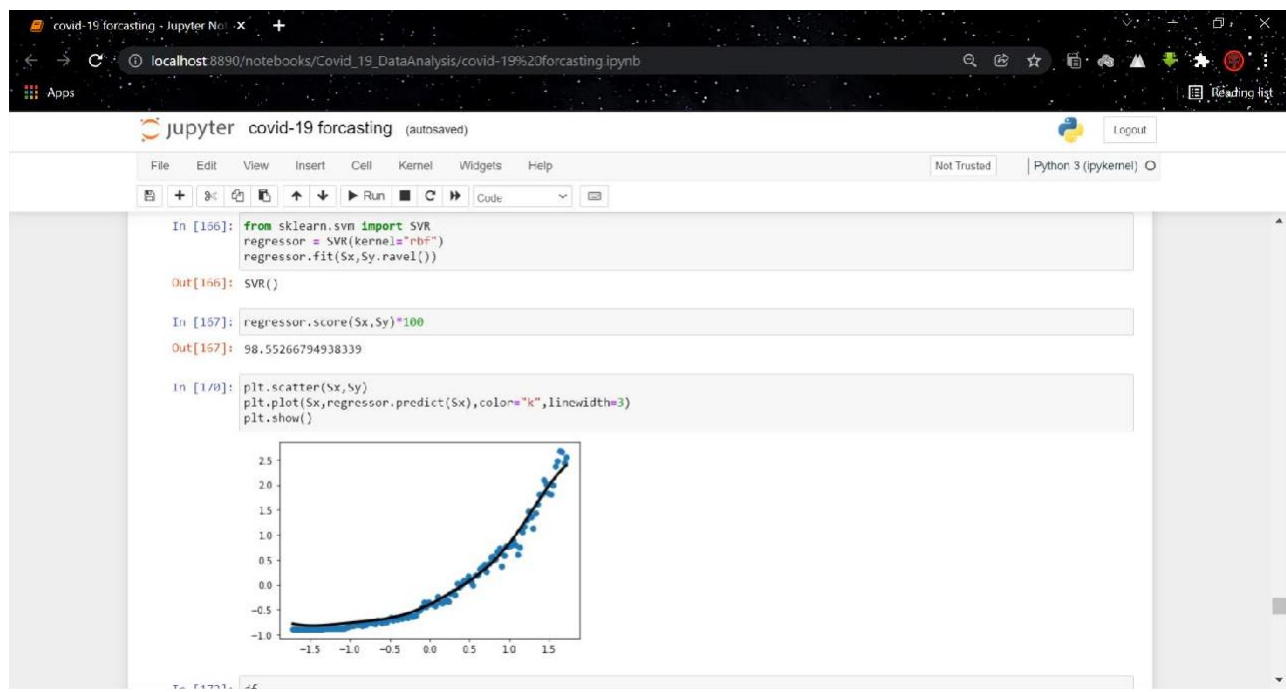


The screenshot shows a Jupyter Notebook interface for 'covid-19 forecasting'. The browser address bar indicates the notebook is running on localhost:8890. The notebook title is 'covid-19 forecasting (autosaved)'. The interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Widgets, Help) and a toolbar with icons for running, saving, and other actions. The kernel is identified as 'Python 3 (pykernel)'. The notebook content consists of two code cells. The first cell (In [164]) imports StandardScaler from sklearn.preprocessing and applies it to data X and y, resulting in transformed data Sx and Sy. The second cell (In [171]) displays the output of Sx, which is a 15x2 array of standardized values.

```
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)
```

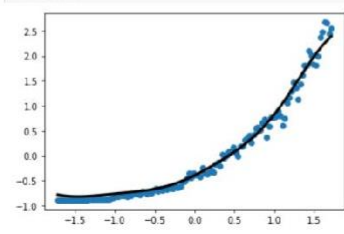
```
In [171]: Sx
Out[171]: array([[ -1.71813853],
 [ -1.69020132],
 [ -1.6622641 ],
 [ -1.63432689],
 [ -1.60638968],
 [ -1.57845247],
 [ -1.55051526],
 [ -1.52257804],
 [ -1.49464083],
 [ -1.46670362],
 [ -1.43876641],
 [ -1.4108292 ],
 [ -1.38289199],
 [ -1.35495477],
 [ -1.32701756],
 [ -1.29908035],
 [ -1.27114314],
 [ -1.24320593],
 [ -1.21526871],
```

```
In [166]: from sklearn.svm import SVR
regressor = SVR(kernel='rbf')
```



The screenshot shows the same Jupyter Notebook interface. The third code cell (In [166]) imports SVR from sklearn.svm and fits the model to the transformed data Sx and Sy. The output (Out [166]) is 'SVR()'. The fourth code cell (In [167]) calculates the score of the model on the training data, resulting in 98.55266794938339. The fifth code cell (In [170]) creates a scatter plot of the original data (Sx, Sy) and overlays a kernel regression line (color='k', linewidth=3). The plot shows a clear upward trend with some scatter, and the regression line follows the general curve of the data points.

```
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 scatter plot displays the relationship between the transformed input variable Sx (x-axis, ranging from -1.5 to 1.5) and the transformed output variable Sy (y-axis, ranging from -1.0 to 2.5). The data points are represented by blue dots, showing a strong positive correlation. A solid black line represents the kernel regression fit, which smoothly follows the upward trend of the data points.

# Covid-19 Data Analysis

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

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

```
Out[172]:
```

	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 (Mayapuri 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	Udara and Nagar Haveli	Udara and Nagar Haveli and Daman and Diu	Hospitalized	07	07	2020
22791	0.0	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()
```

```
Out[173]:
```

Month	Day	Num Cases
04	05	5.0
06	10	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 [174]: x = np.arange(len(Day))
```

```
Out[174]: array([ 0, ...])
```

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

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

```
Out[173]:
```

Month	Day	Num Cases
04	05	5.0
06	10	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 [174]: x = np.arange(len(Day))
```

```
Out[174]: array([ 0, ...])
```



# Covid-19 Data Analysis

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

jupyter covid-19 forecasting (autosaved)

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

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])
```

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

jupyter covid-19 forecasting (autosaved)

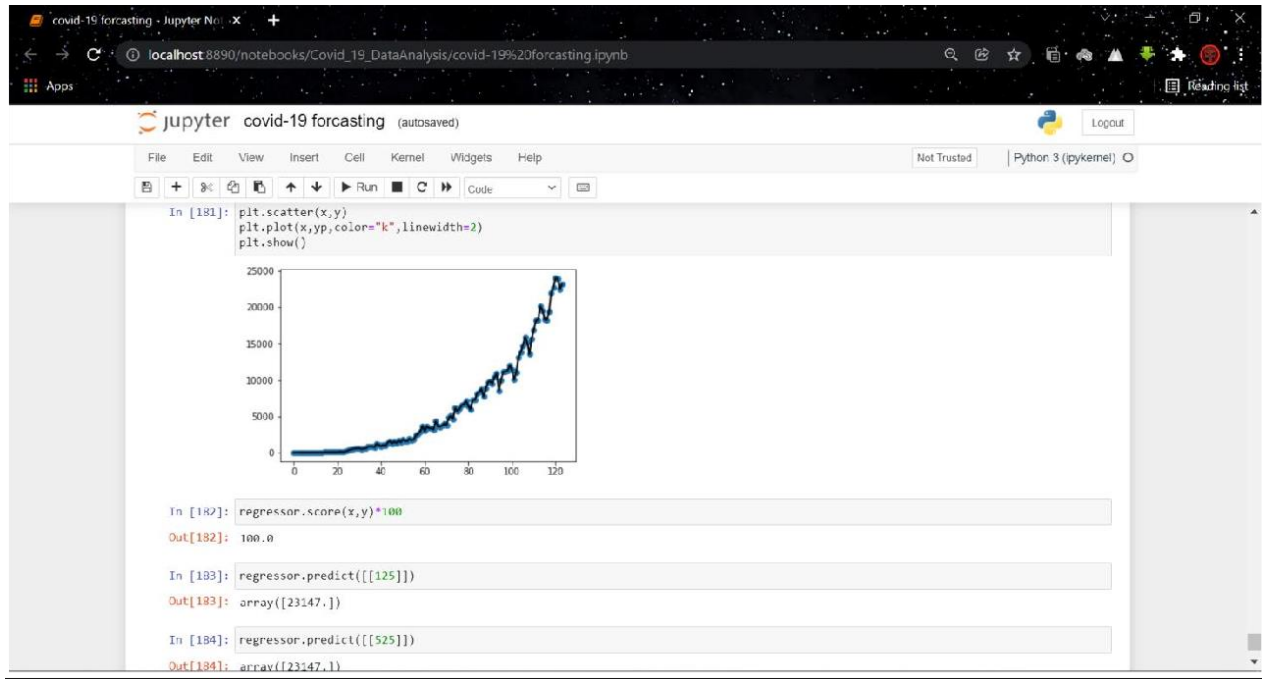
File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

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, 1.6000e+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, 5.3000e+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, 1.0313e+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, 3.9913e+03,
                3.8080e+03, 4.7940e+03, 5.0450e+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,
                9.0470e+03, 9.4730e+03, 1.0460e+04, 1.0992e+04, 9.5260e+03])
```

# 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.

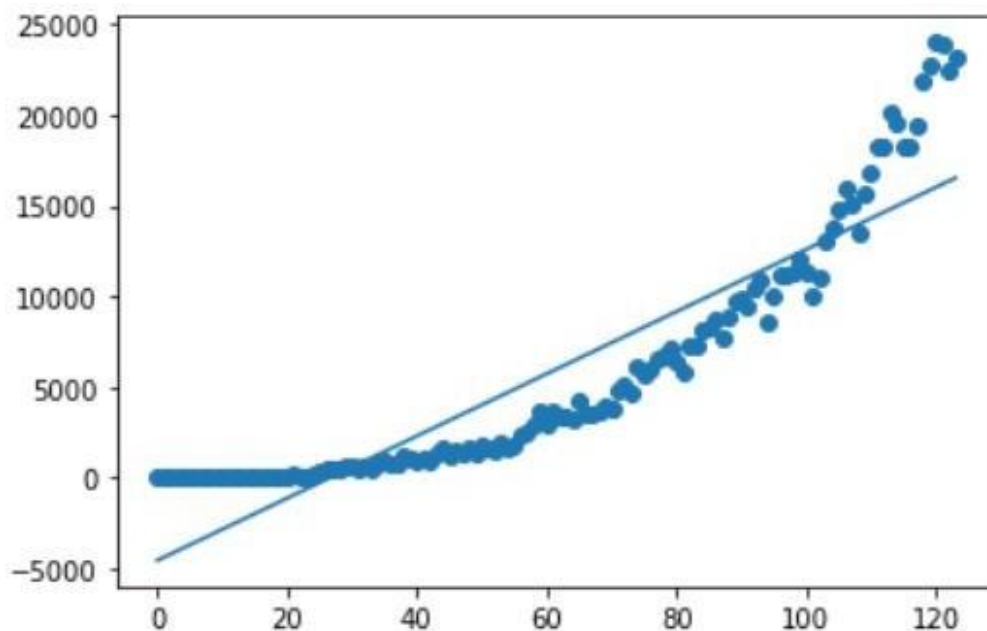
## Results and Discussion

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.

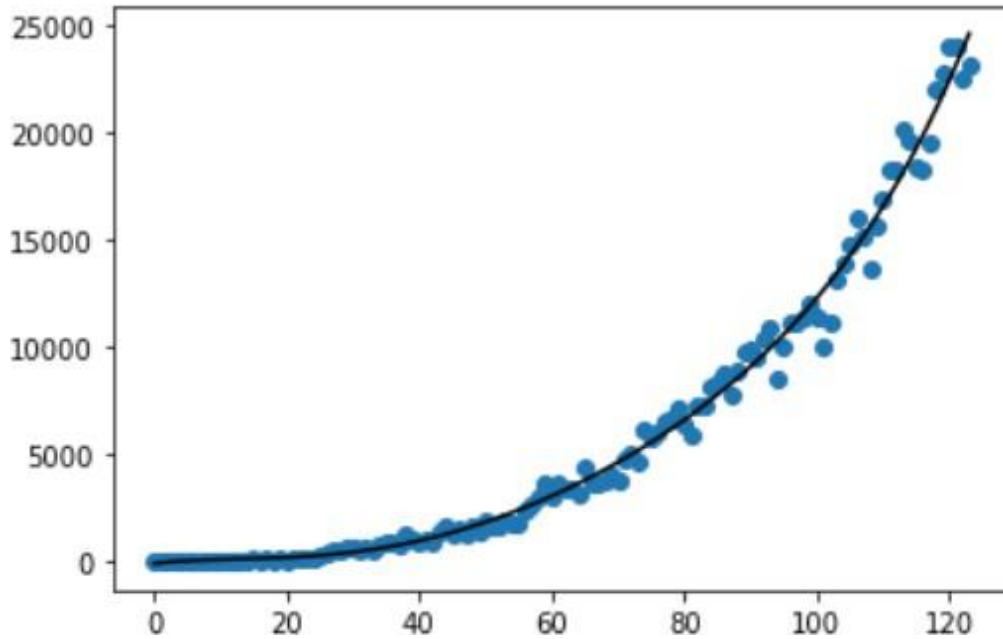
What we can see here is that number of affected males is double than the no. of females affected. This means that males have double the chance of contracting covid-19 than a female.

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.

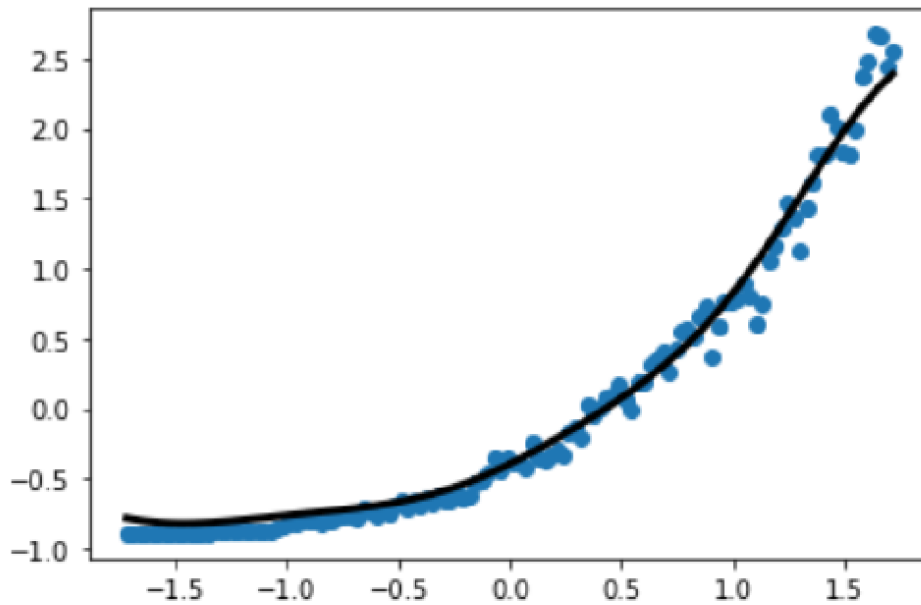
The diagram shows the prediction line using linear regression for cases that was finally presented.



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.

## **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.





