

# PIMPRI CHINCHWAD COLLEGE OF ENGINEERING

A project report on

Covid 19 Data Analysis

**Student Name** **Mentor Name**

Sujeet Jawale Prof. Sachin Lohar

PCET Knowledge Solutions India

* **TABLE OF CONTENTS:**

|  |  |
| --- | --- |
| 1. | TABLE OF FIGURES |
| 2. | ABSTRACT |
| 3. | CHAPTERS |
| 4. | COMPLETE CODE |
| 5. | CONCLUSION |

**TABLE OF FIGURES**

|  |
| --- |
| **1. STATEWISE COVID DATA IN INDIA**  Figure 1…………………………………………….15  Figure 2…………………………………………….15  Figure 3…………………………………………….16  Figure 4…………………………………………….16  Figure 5…………………………………………….17  **2. OVERALL COVID 19 STATUS IN INDIA**  Figure 6…………………………………………….23  **3. VISUALISING THE SPREAD GEOGRAPHYLLY**  Map 1………………………………………………30  Map 2………………………………………………31  Map 3………………………………………………32  **4. TREND OF VIRUS**  Graph 1…………………………………………….34  Graph 2…………………………………………….34  Graph 3…………………………………………….35  Graph 4…………………………………………….36  **5. FORCAST USING fbprophet**  Graph 1…………………………………………….40  Graph 2…………………………………………….41  Graph 3…………………………………………….43 |

**ABSTRACT**

We are creating a data analysis project on spread of novel coronavirus in India the project use raw data in form of .csv and .xls files and transform it into data analysis. This project is an attempt of Data Analysis Coronavirus covid-19 is spread in India with the help of data science and Data Analytics in python code. This data will help us to find the basis behind common notions about the virus spread from purely a data set perspective.

The data used for this project is split across two files :

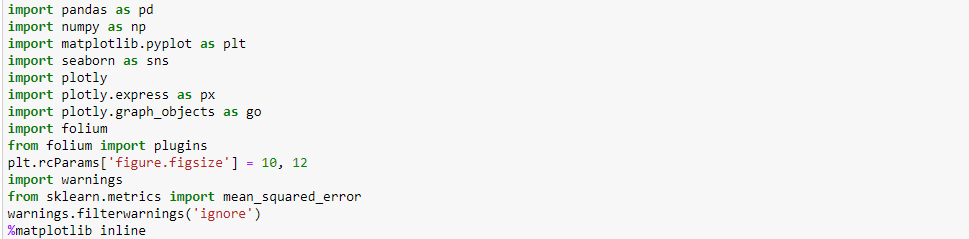
1. covid\_19\_india.csv

2. Indian\_coordinates.xls

For the analysis we use some libraries such as Pandas, numpy, matplotlib, seaborn, plotly, stdlib, folium, warnings and fbprophets in the project we import files by using user defined functions we clean the data. In the first part of the workshop, we will use Python libraries such as Pandas and Matplotlib to download, clean, analyze and visualize the coronavirus open dataset. And after that we are going to visualise the spread geographically we use Folium. And then with scatter plot, line plot and bar plot we have visualized the data in graphs. And at the end Fbprophet is used for visualization using graphs and proper visualization is done.

**CHAPTERS**

IMPORTING THE DATA VISUALIZATION LIBRARIES IN PYTHON :



- Import pandas as pd : First we need to install all these libraries and then we can import and use them in our code we import pandas as pd simply imports the libraries the current namespace , but rather than using the name as pandas it is called as pd. Similarly we have to import all the libraries Numpy, pyplot, seaborn, folium , sklearn

- Import Numpy as np : NumPy offers comprehensive mathematical functions, random number generators, linear algebra routines, Fourier transforms, and more.

- Import Matplotlib : Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Basically it is used for visualization of data with the help of graphs and plots.

- Import plotly : The [plotly Python library](https://plotly.com/python/) is an interactive, [open- source](https://plotly.com/python/is-plotly-free) plotting library that supports over 40 unique chart types covering a wide range of statistical, financial, geographic, scientific, and 3-dimensional use-cases.

- Import plotly.express as px : Plotly Express is a terse, consistent, high-level API for creating figures.

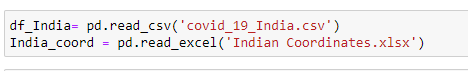
- Import plotly.graph\_object as go : **he**plotly.graph\_objects**module (typically imported as**go**) contains an**[automatically-generated hierarchy of Python classes](https://plotly.com/python-api-reference/plotly.graph_objects.html#graph-objects)**which represent non-leaf nodes in this figure schema. The term "graph objects" refers to instances of these classes.**

**-Import Folium :** folium builds on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js library. Manipulate your data in Python, then visualize it in a Leaflet map via folium.

- Import Warnings : Warnings are provided to warn the developer of situations that aren’t necessarily exceptions. Usually, a warning occurs when there is some obsolete of certain programming elements, such as keyword, function or class.

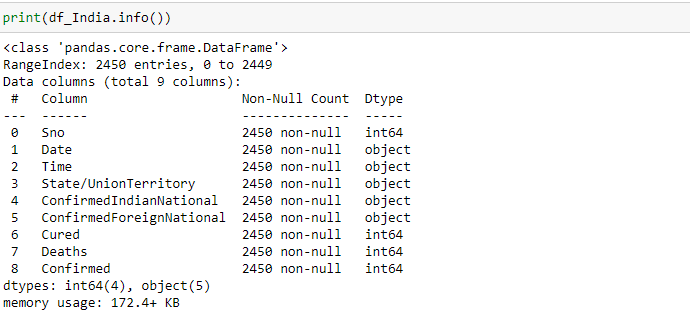
- From sklearn.metrices import mean\_square\_error : **Mean Squared Deviation** of an estimator measures the average of error squares i.e. the average squared difference between the estimated values and true value. It is a risk function, corresponding to the expected value of the squared error loss. It is always non – negative and values close to zero are better.

READING THE CSV FILE AND XLS FILE :



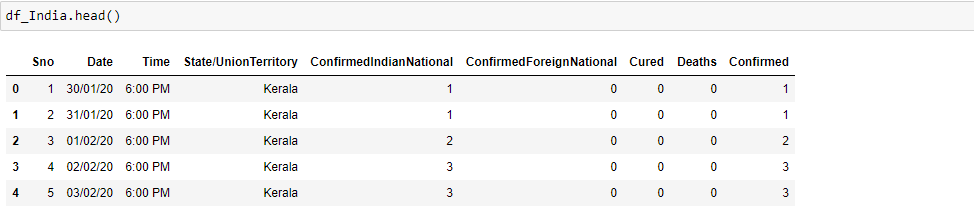
* df\_India = pd.read\_csv(‘’) : this is the line of code where the csv file is loaded and read.
* India\_coord = pd.read\_excel(‘’) : this is the line of code where the xls file is loaded and read.

.INFO FUNCTION :



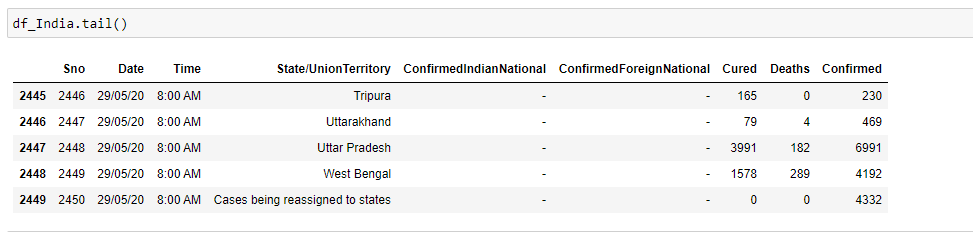
.info() : Is a inbuilt function that is used to take out an outlook of data it tells us the structure of data and also the dtypes how many objects are used and memory usage.

.HEAD FUNCTION :



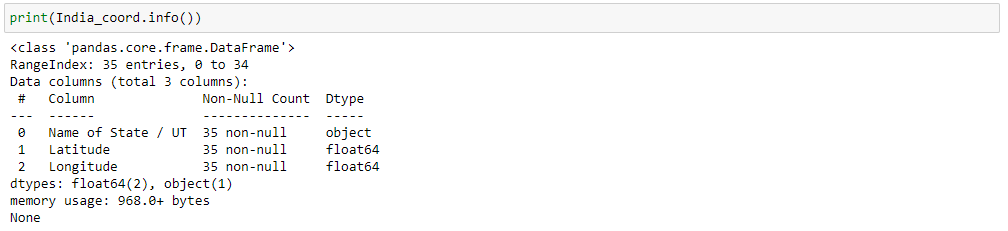
-.head() : Is a inbuilt function that is used to take out an outlook first 5 rows of the data set .

.TAIL FUNCTION :



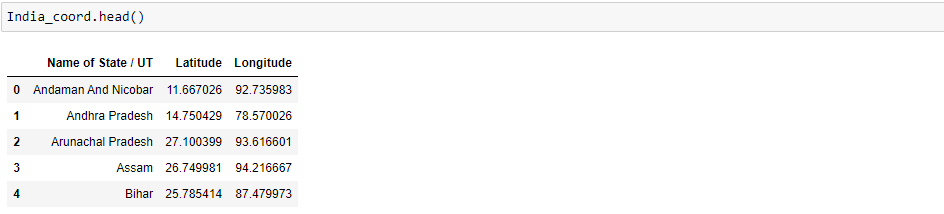
-.tail() : Is a inbuilt function that is used to take out an outlook last 5 rows of the data set .

.INFO FUNCTION :



-.info() : Is a inbuilt function that is used to take out an outlook of data it tells us the structure of data and also the dtypes how many objects are used and memory usage.

.HEAD FUNCTION :



-.head() : Is a inbuilt function that is used to take out an outlook first 5 rows of the data set .

CLEANING OF DATASET :

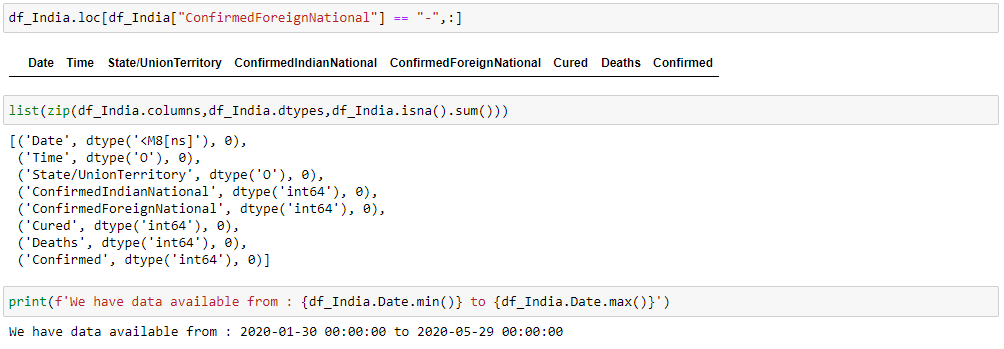


Here we create a function named as replace\_dash\_with\_zero() and with the help of this function we are going to replace all the missing values in the dataset which are given as ’ – ‘ will be replaced with 0

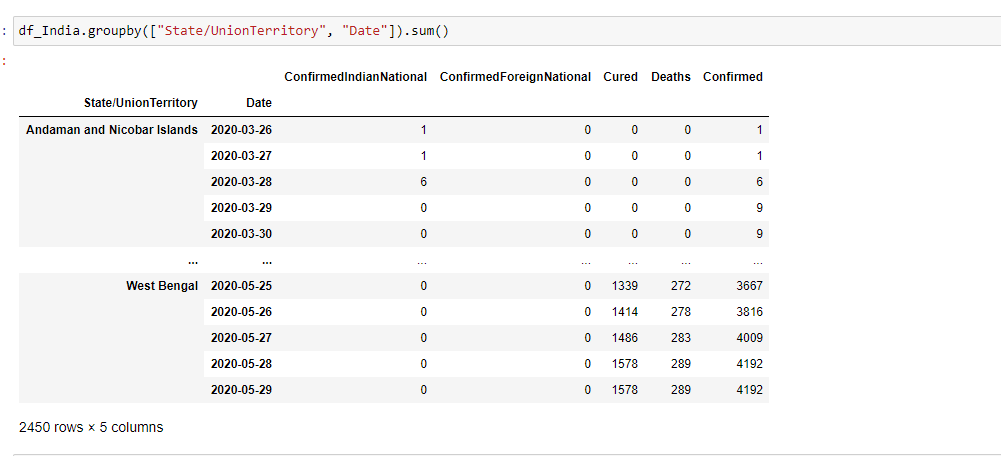
df\_india\_drop function : drops the serial number column as it is an unnecessary column in the dataset.

df\_India[‘Date’] : this line of code is changing the format of date column to date-month-year.

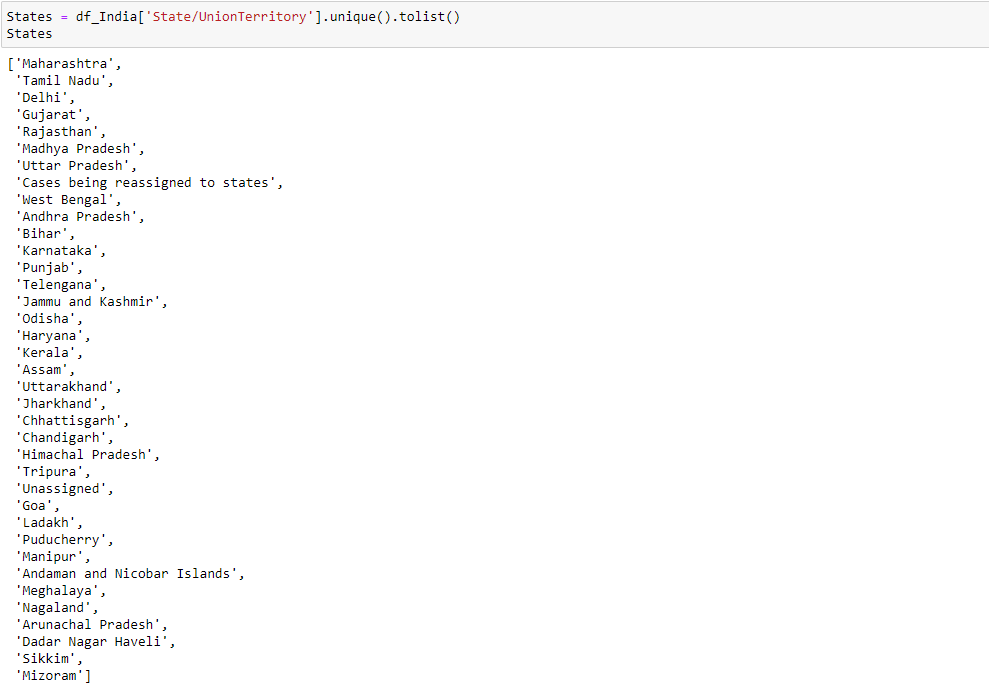
After this the replace\_dash\_with\_zero Function is applied to ConfirmedIndianNational and ConfirmedForigenNational so that the dash will be replaced with the 0.



Now as we have changed the format of the date to date-month-year then now we have to check that we have the data from min where to max means the data is from this date to this date and then we have to print that date as shown in image.

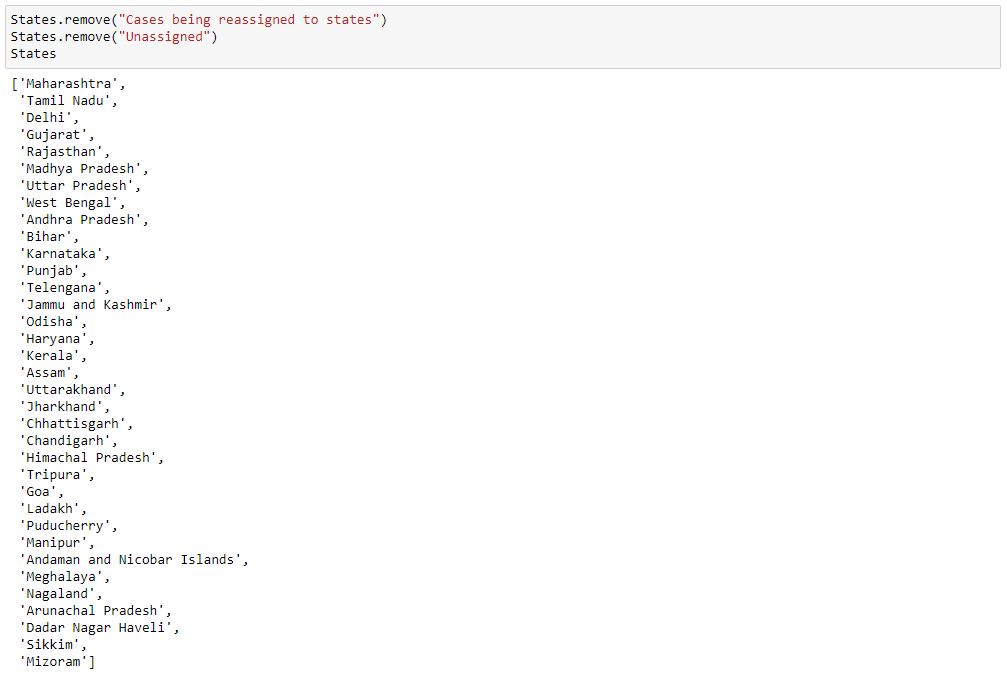


Now here we have used the groupby inbuilt function. A groupby operation involves some combination of splitting the object, applying a function, and combining the results. This can be used to group large amounts of data and compute operations on these groups.

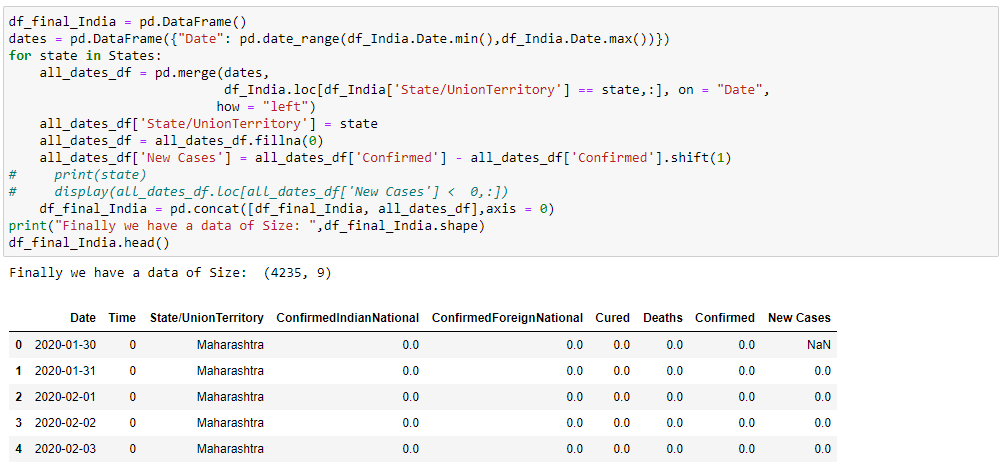


* States = df\_India[‘State/UnionTerritory’].unique().tolist() :

In this we will list down the unique state/territory into and list and then display that list unique Function. The unique() function is used to find the unique elements of an array. Returns the sorted unique elements of an array. ... the indices of the unique array that reconstruct the input array. the number of times each unique value comes up in the input array. And hence we get list of different states in the dataset. Pandas tolist() is used to convert a series to list. Initially the series is of type pandas.core.series.Series and applying tolist() method, it is converted to list data type.



Remove() : **remove()** is an inbuilt function in Python programming language that removes a given object from the list. It does not return any value. Here wehave removed the case that is being ressigned to states and the unassigned rows or states.



df\_final\_India = pd.Dataframe() :

Data structure also contains labeled axes (rows and columns). Arithmetic operations align on both row and column labels. Can be thought of as a dict-like container for Series objects. The primary pandas data structure.

Dates = pd.Dataframe() :

Here we are selecting all the rows with the help of dates column and applying for loop to it for further analysis.

For Loop ;

In the for loop for state in states we are using merge function to merge the dates and the loc df . and for all dates we are giving a name ‘state’ and if there is any empty column then we are using fillna() function to fill the empty space as 0.

Fill na() : Pandas is one of those packages, and makes importing and analyzing data much easier. Sometimes csv file has null values, which are later displayed as NaN in Data Frame. Just like pandas dropna() method manage and remove Null values from a data frame, fillna() manages and let the user replace NaN values with some value of their own.

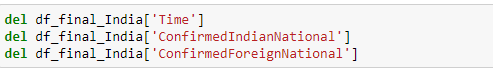
DROPNA() :



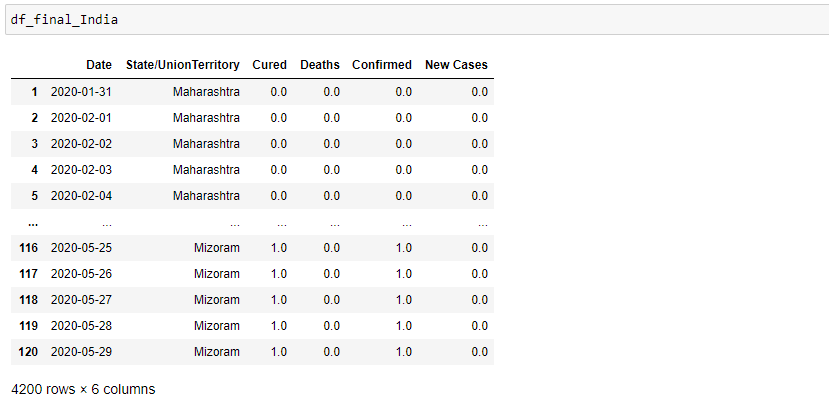
Df\_final\_India.dropna() :

Pandas is one of those packages and makes importing and analyzing data much easier. Sometimes csv file has null values, which are later displayed as NaN in Data Frame. Pandas dropna() method allows the user to analyze and drop Rows/Columns with Null values in different ways.

DELETE :

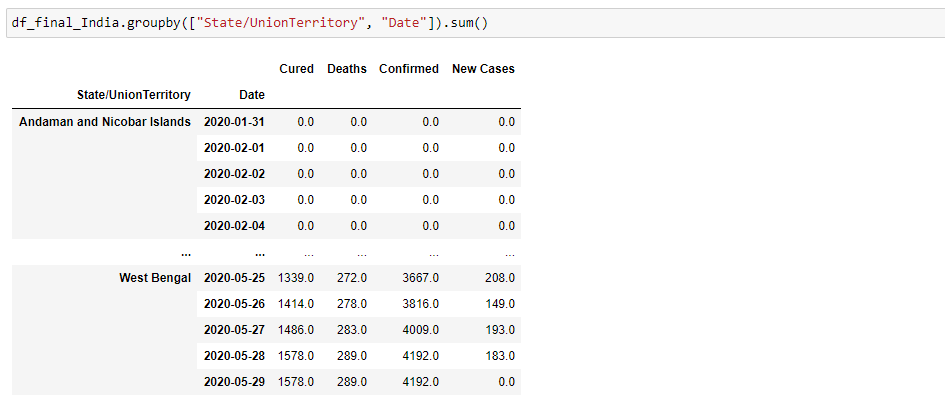


del : after this we delete three columns which are not further required Time, ConfirmedIndianNational and ConfirmedForigenNational with the help of del .

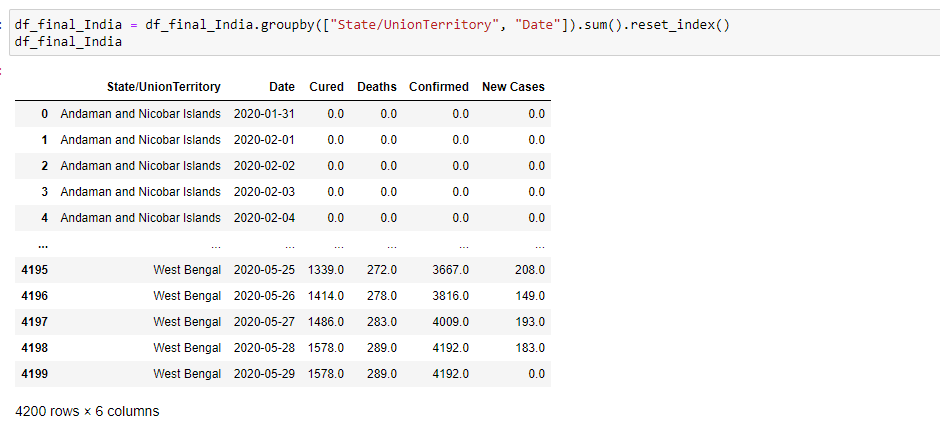


Displaying the Dataset by calling it by its name df\_final\_India.

SUM() :

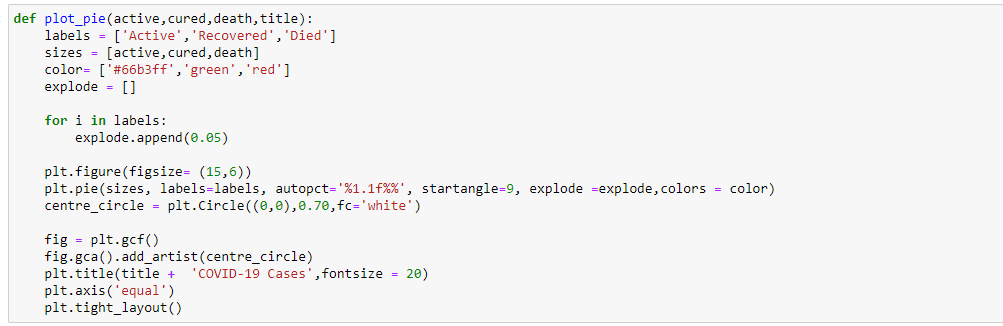


sum() : Sum of numbers in the list is required everywhere. Python provide an inbuilt function sum() which sums up the numbers in the list. And hence we line up the data in dataset with reference to the state and date and last we calculate. Sum of numbers in the list is required everywhere. Python provide an inbuilt function sum() which sums up the numbers in the list

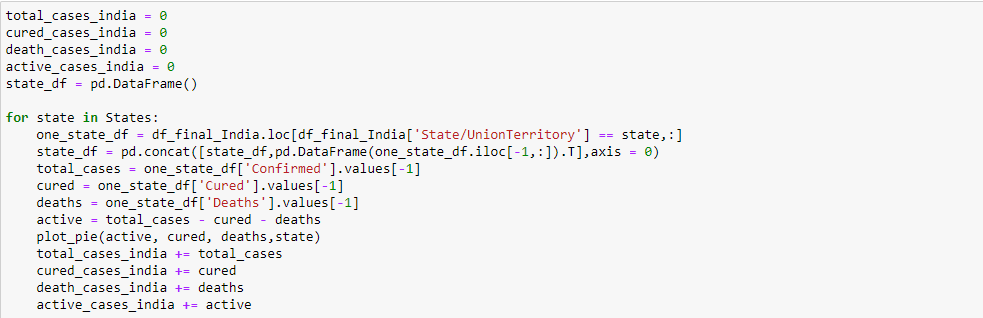


Sum().reset\_index() : Often We start with a huge dataframe in Pandas and after manipulating/filtering the dataframe, we end up with much smaller dataframe. When we look at the smaller dataframe, it might still carry the row index of the original dataframe. If the original index are numbers, now we have indexes that are not continuous. Well, pandas has [reset\_index()](https://www.geeksforgeeks.org/python-pandas-dataframe-reset_index/) function. So to reset the index to the default integer index beginning at 0, We can simply use the reset\_index() function.

So from here the index is set to the default integer index beginning at 0 hence we use reset\_index().

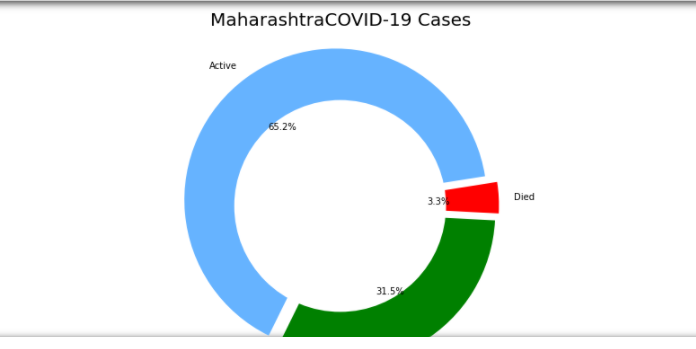


Here we create a fuction named as plot\_pie in which three arguments are passed active, cured, and death and title. Then a for loops which appends 0.05 after each iteration now form here we are going towards the visualization part so with the help of plt.figure we the shape to the pie and with pie we give lables to the pie . after that with the title we give title as the name of each state and As we have to visualize all the states we give the points and give an specific layout now we are ready.

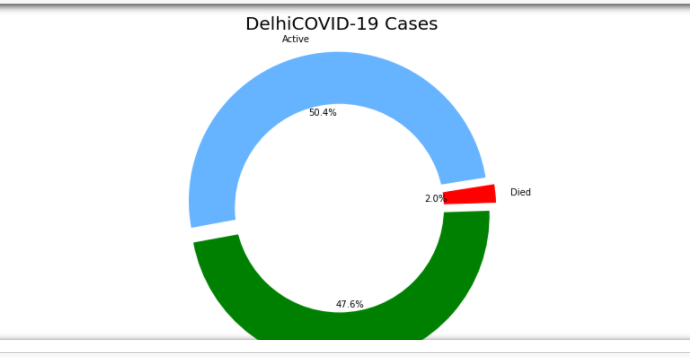


For visualization through the pie plot we now need some more things,

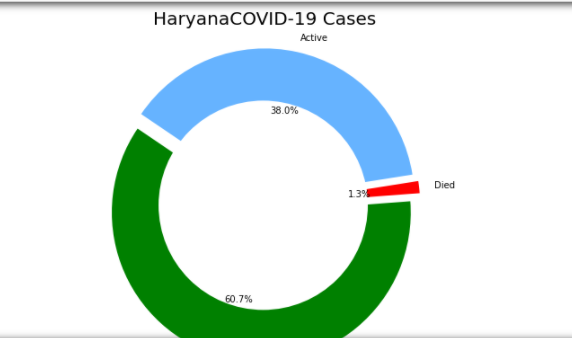
Now we specify some variables so that we can use them further while plotting the pie. A **Pie Chart**is a circular statistical plot that can display only one series of data. The area of the chart is the total percentage of the given data. The area of slices of the pie represents the percentage of the parts of the data. The slices of pie are called wedges. The area of the wedge is determined by the length of the arc of the wedge. The area of a wedge represents the relative percentage of that part with respect to whole data. Pie charts are commonly used in business presentations like sales, operations, survey results, resources, etc as they provide a quick summary.



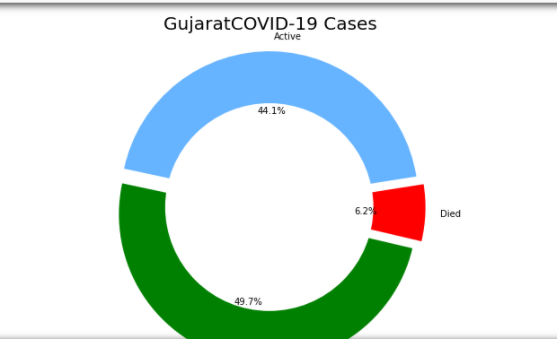
Pie chart For Maharashtra State



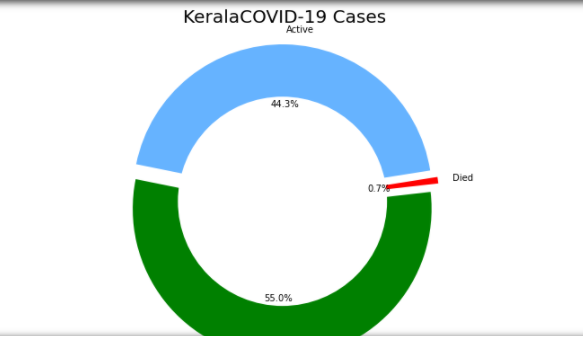
Pie Chart for Delhi State



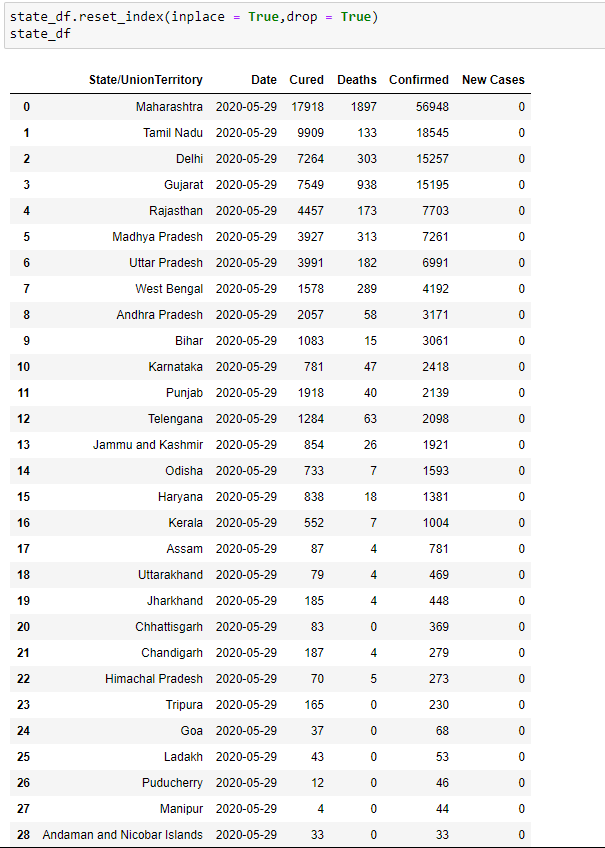
Pie Chart For Haryana State



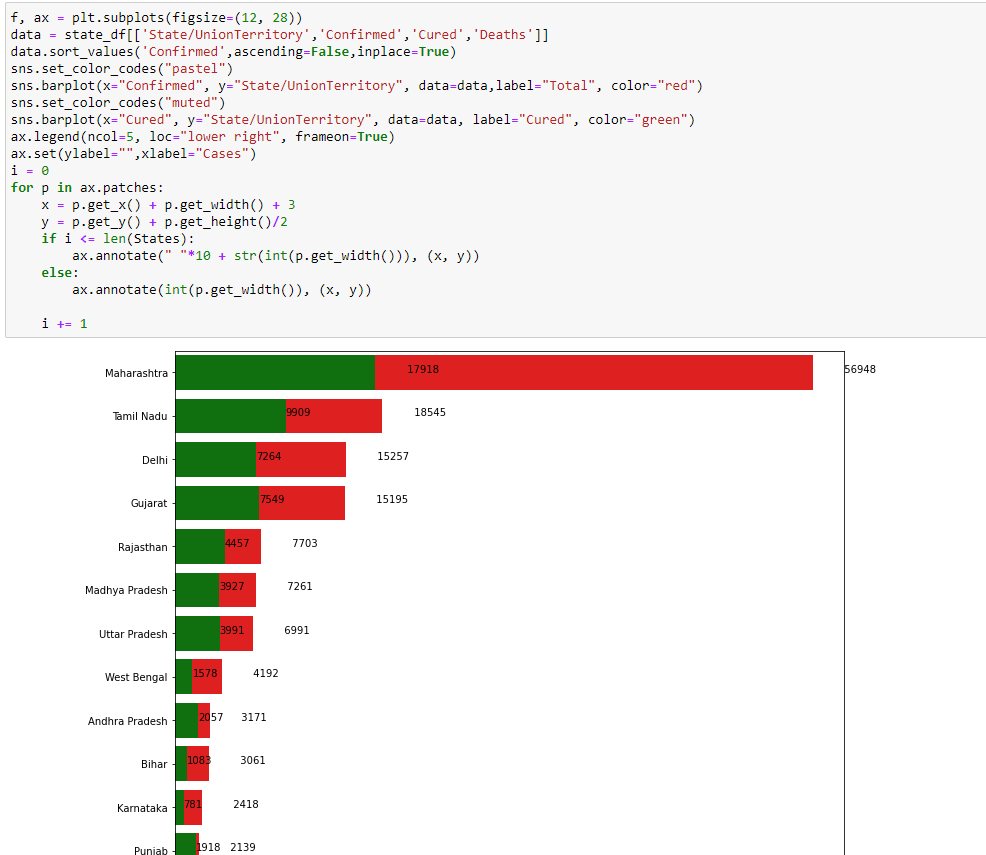
Pie Chart for Gujarat State



Pie Chart for Kerala State



After Ploting the pie now we again reset the index by using reset\_index() with its two attributes inplace and drop. Now as we know again the indexing will go to 0. reset\_index() , Often We start with a huge dataframe in Pandas and after manipulating/filtering the dataframe, we end up with much smaller dataframe. When we look at the smaller dataframe, it might still carry the row index of the original dataframe. If the original index are numbers, now we have indexes that are not continuous. Well, pandas has [reset\_index()](https://www.geeksforgeeks.org/python-pandas-dataframe-reset_index/) function. So to reset the index to the default integer index beginning at 0, We can simply use the reset\_index() function.



BAR PLOT :

A bar chart or bar graph is a chart or graph that presents categorical data with rectangular bars with heights or lengths proportional to the values that they represent. The bars can be plotted vertically or horizontally.

A bar graph shows comparisons among discrete categories. One axis of the chart shows the specific categories being compared, and the other axis represents a measured value.

Matplotlib API provides the bar() function that can be used in the MATLAB style use as well as object oriented API. The signature of bar() function to be used with axes.

Here we will do bar plot to visualize data in deep and get some more instints from it.

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

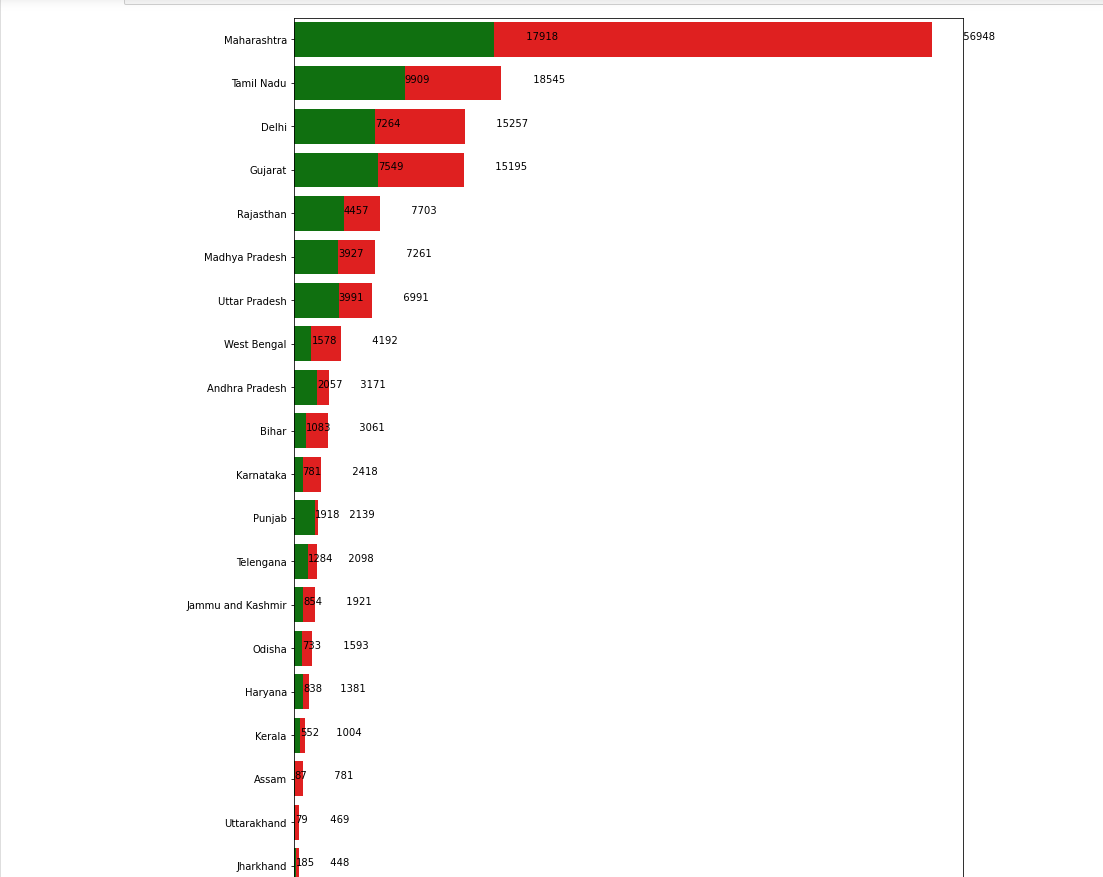
plt.subplots() function you can draw multiple plots in one figure.so we use this and give it the figsize which we want to give.

Then the data is sorted using sort\_value function and then color code as ‘pastel’.

And after that sns.barplot() to plot the graph, and the color code is set to muted.

Then we use a for loop and iterate p in the ax.patches : and set x = p.get\_x() + p.get\_width() +3 and y = p.get\_y() + p.get\_height()/2 .

Then an if loop is used where the length of states is less than the states, and use annotate() function n the if loop if case satisfy the condition then ax.annotate(), else ax.annotate(int(p.get\_width()),(x,y)).



Here we will do bar plot to visualize data in deep and get some more instints from it.

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

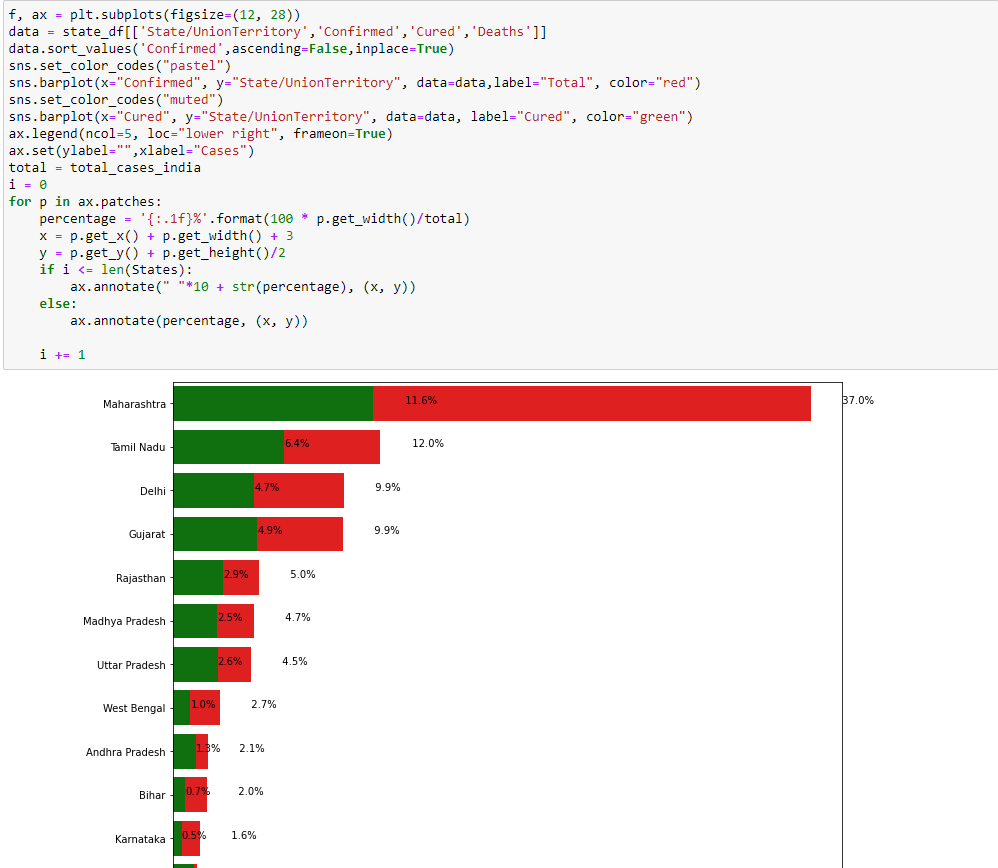
plt.subplots() function you can draw multiple plots in one figure.so we use this and give it the figsize which we want to give.

Then the data is sorted using sort\_value function and then color code as ‘pastel’.

And after that sns.barplot() to plot the graph, and the color code is set to muted.

Then we use a for loop and iterate p in the ax.patches : and set x = p.get\_x() + p.get\_width() +3 and y = p.get\_y() + p.get\_height()/2 .

Then an if loop is used where the length of states is less than the states, and use annotate() function n the if loop if case satisfy the condition then ax.annotate(), else ax.annotate(int(p.get\_width()),(x,y)).



Bar plot to visualize data in deep and get some more instints from it.

A bar plot or bar chart is a graph that represents the category of data with rectangular bars with lengths and heights that is proportional to the values which they represent. The bar plots can be plotted horizontally or vertically. A bar chart describes the comparisons between the discrete categories. One of the axis of the plot represents the specific categories being compared, while the other axis represents the measured values corresponding to those categories.

plt.subplots() function you can draw multiple plots in one figure.so we use this and give it the figsize which we want to give.

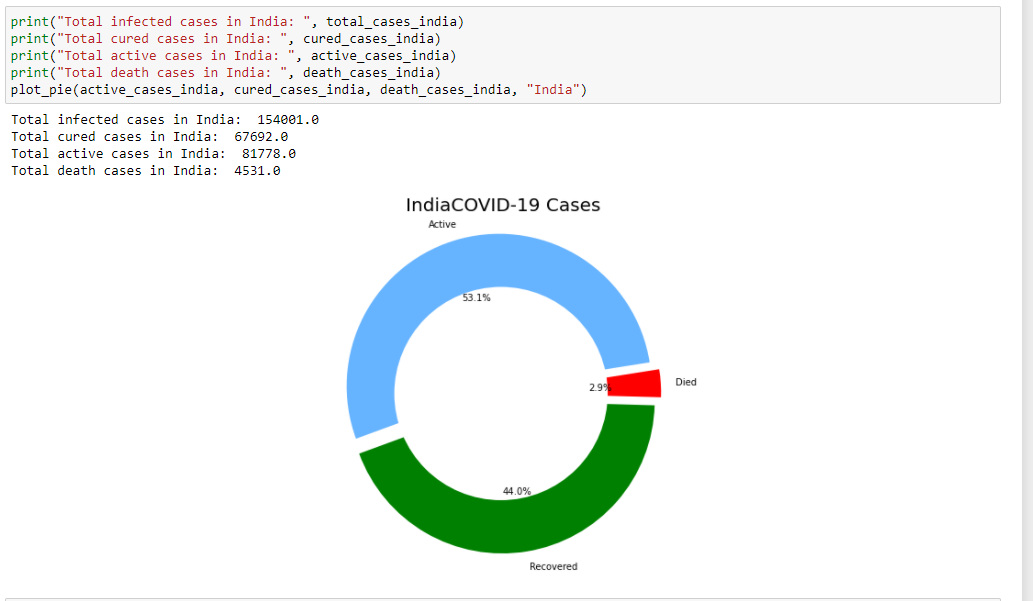
Then the data is sorted using sort\_value function and then color code as ‘pastel’.

And after that sns.barplot() to plot the graph, and the color code is set to muted.

Then we use a for loop and iterate p in the ax.patches : and set x = p.get\_x() + p.get\_width() +3 and y = p.get\_y() + p.get\_height()/2 .

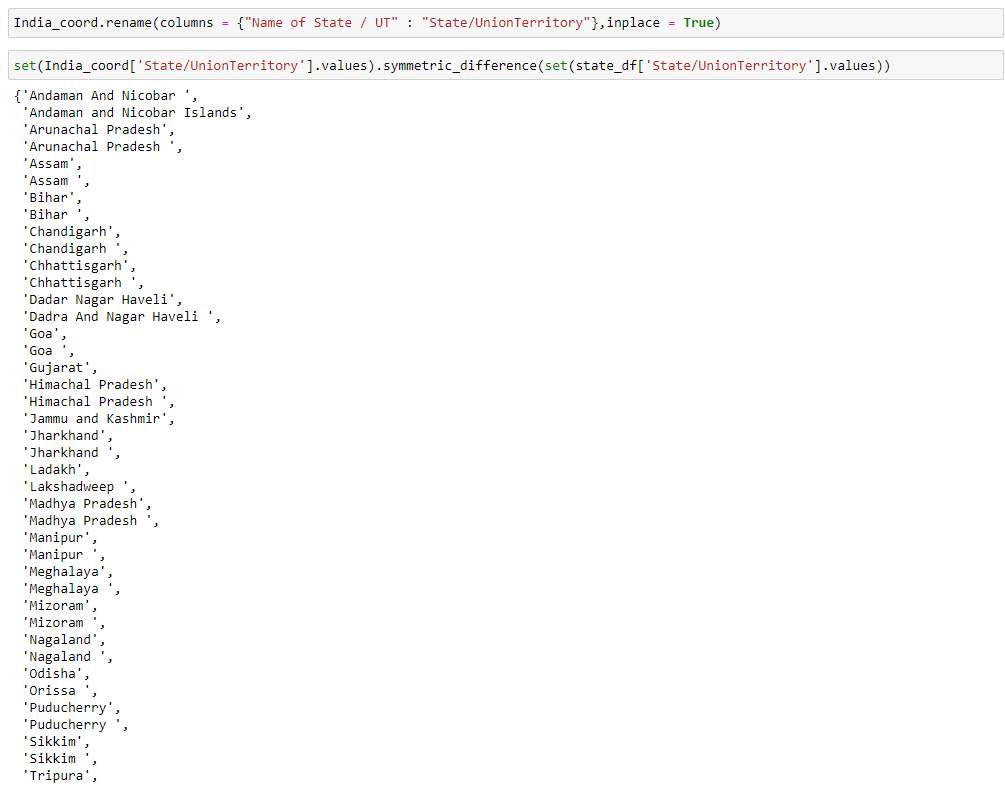
Then an if loop is used where the length of states is less than the states, and use annotate() function n the if loop if case satisfy the condition then ax.annotate(), else ax.annotate(int(p.get\_width()),(x,y)). PEP stands for Python Enhancement Proposal. It is a design document that describes new features for Python or its processes or environment. It also provides information to the python community. PEP is a primary mechanism for proposing major new features, for example – Python Web Server Gateway Interface, collecting the inputs of the community on the issues and documenting design decisions that have been implemented in Python. Function annotations are arbitrary python expressions that are associated with various part of functions. These expressions are evaluated at compile time and have no life in python’s runtime environment. Python does not attach any meaning to these annotations. They take life when interpreted by third party libraries, for example, mypy.

**OVERALL COVID 19 STATUS IN INDIA**



Now we want to visualize for whole India nit just for the States so we call all the variables such as total\_cases\_india, cured\_cases\_india, active\_case\_india, and Death\_cases\_india. So that after that we can plot the pie chart and visualize for whole India so after that we call the function plot\_pie() and give it the attributes and it displays the pie chart.

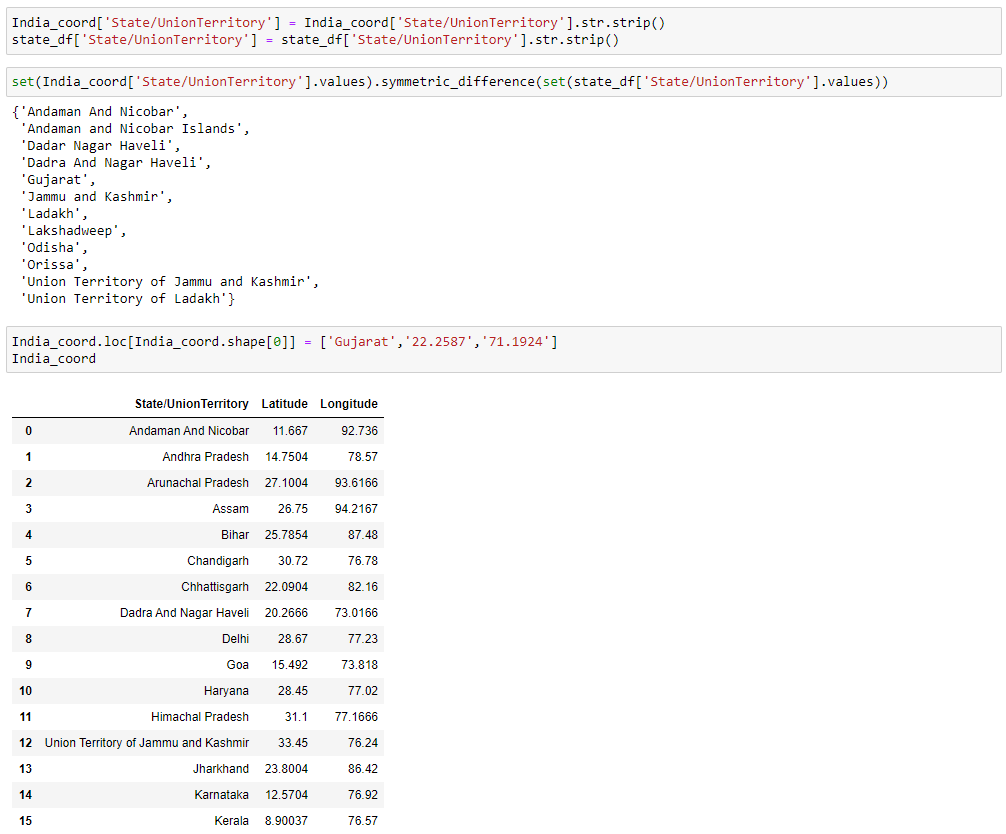
**VISUALISING THE SPREADS GEOGRAPHICALLY**



India\_coord.rename(columns = {"Name of State / UT" : "State/UnionTerritory"},inplace = True)

Here we rename the name of columns with the help of rename function and do the further operation on it. In Python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. os***.***rename() method in Python is used to rename a file or directory. This method renames a source file/ directory to specified destination file/directory.

set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values)) : **values()** is an inbuilt method in Python programming language that returns a list of all the values available in a given dictionary.



India\_coord.rename(columns = {"Name of State / UT" : "State/UnionTerritory"},inplace = True)

Here we rename the name of columns with the help of rename function and do the further operation on it. In Python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. os***.***rename() method in Python is used to rename a file or directory. This method renames a source file/ directory to specified destination file/directory.

set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values)) : **values()** is an inbuilt method in Python programming language that returns a list of all the values available in a given dictionary.

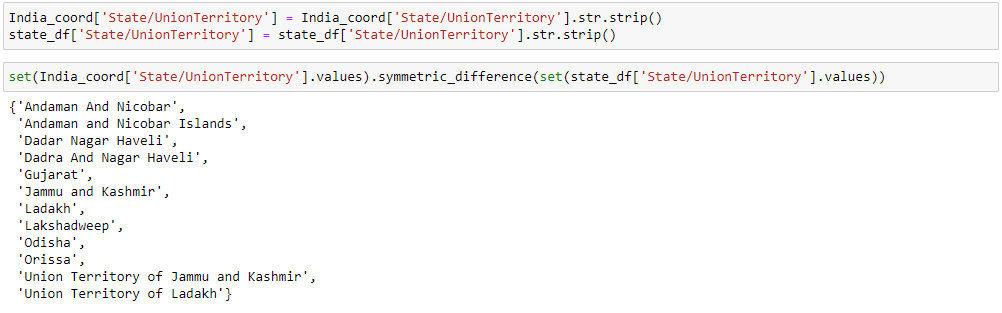
AFTER THIS WE,

India\_coord['State/UnionTerritory'] = India\_coord['State/UnionTerritory'].str.strip()

state\_df['State/UnionTerritory'] = state\_df['State/UnionTerritory'].str.strip()

set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values))

**strip()** is an inbuilt function in Python programming language that returns a copy of the string with both leading and trailing characters removed (based on the string argument passed). We use strip function.



India\_coord.rename(columns = {"Name of State / UT" : "State/UnionTerritory"},inplace = True)

Here we rename the name of columns with the help of rename function and do the further operation on it. In Python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. os***.***rename() method in Python is used to rename a file or directory. This method renames a source file/ directory to specified destination file/directory.

set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values)) : **values()** is an inbuilt method in Python programming language that returns a list of all the values available in a given dictionary.

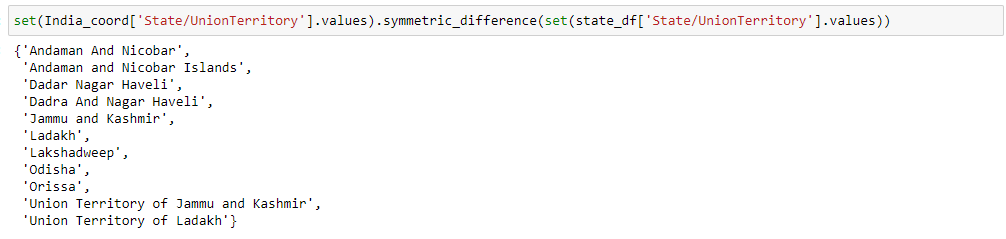
AFTER THIS WE,

India\_coord['State/UnionTerritory'] = India\_coord['State/UnionTerritory'].str.strip()

state\_df['State/UnionTerritory'] = state\_df['State/UnionTerritory'].str.strip()

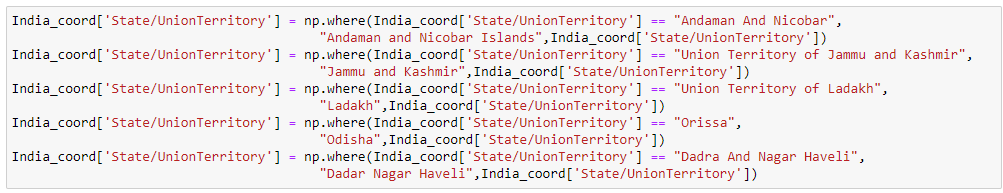
set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values))

**strip()** is an inbuilt function in Python programming language that returns a copy of the string with both leading and trailing characters removed (based on the string argument passed). We use strip function.



Here we set india coordinate statewise value and use symmetric\_difference function.

This in-built function of Python Set helps us to get the symmetric difference between two sets, which is equal to the**elements present in either of the two sets, but not common to both the sets**. Let’s look at the Venn diagram of the symmetric\_difference between two sets. This in-built function of Python Set helps us to get the symmetric difference between two sets, which is equal to the **elements present in either of the two sets, but not common to both the sets.** Let’s look at the Venn diagram of the symmetric\_difference between two sets.



* India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Andaman And Nicobar",

"Andaman and Nicobar Islands",India\_coord['State/UnionTerritory'])

* India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Union Territory of Jammu and Kashmir",

"Jammu and Kashmir",India\_coord['State/UnionTerritory'])

* India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Union Territory of Ladakh",

"Ladakh",India\_coord['State/UnionTerritory'])

* India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Orissa",

"Odisha",India\_coord['State/UnionTerritory'])

* India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Dadra And Nagar Haveli",

"Dadar Nagar Haveli",India\_coord['State/UnionTerritory'])

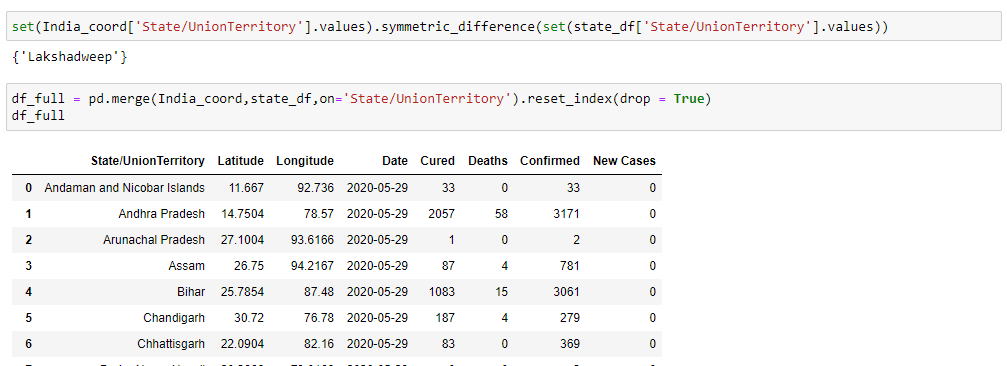
The numpy.where() function returns the indices of elements in an input array where the given condition is satisfied.

Syntax :numpy.where(condition[, x, y])  
Parameters:  
condition : When True, yield x, otherwise yield y.  
x, y : Values from which to choose. x, y and condition need to be broadcastable to some shape.

Returns:  
out : [ndarray or tuple of ndarrays] If both x and y are specified, the output array contains elements of x where condition is True, and elements from y elsewhere.

If only condition is given, return the tuple condition.nonzero(), the indices where condition is True.

Here we use the where function which belong to numpy and give the name of different states separately and for further map and we can show the analysis on the map.



India\_coord.rename(columns = {"Name of State / UT" : "State/UnionTerritory"},inplace = True)

Here we rename the name of columns with the help of rename function and do the further operation on it. In Python provides functions for interacting with the operating system. OS comes under Python’s standard utility modules. This module provides a portable way of using operating system dependent functionality. os***.***rename() method in Python is used to rename a file or directory. This method renames a source file/ directory to specified destination file/directory.

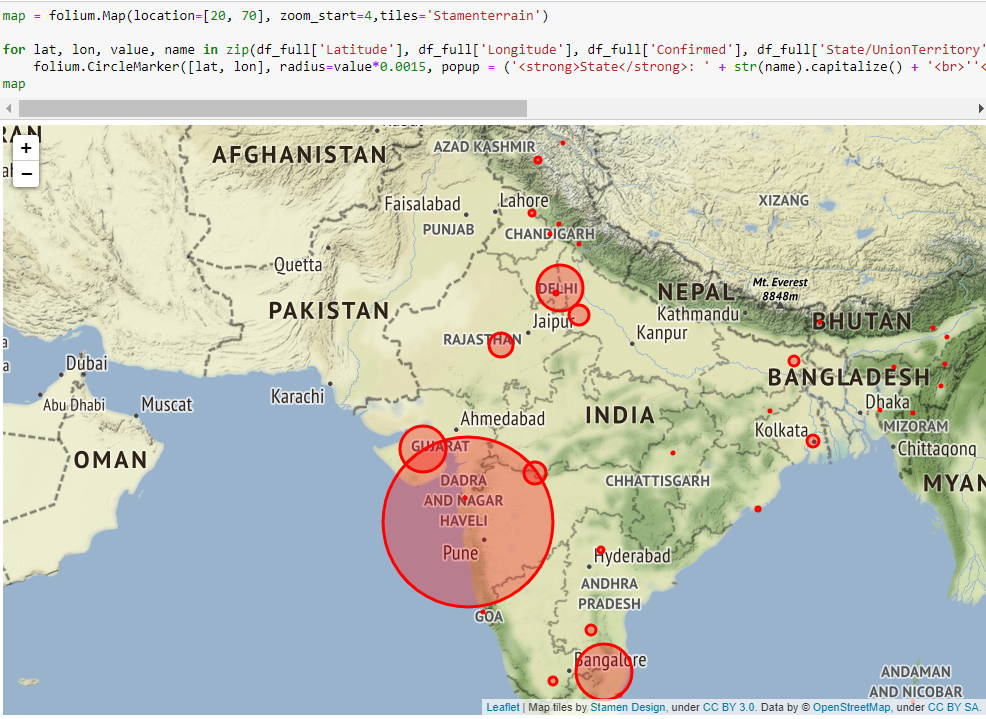
set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values)) : **values()** is an inbuilt method in Python programming language that returns a list of all the values available in a given dictionary.

AFTER THIS WE,

We reset and then merge with the help of inbuilt functions merge() and reset().

df\_full = pd.merge(India\_coord,state\_df,on='State/UnionTerritory').reset\_index(drop = True)

**strip()** is an inbuilt function in Python programming language that returns a copy of the string with both leading and trailing characters removed (based on the string argument passed). We use strip function. [Pandas DataFrame](https://www.geeksforgeeks.org/python-pandas-dataframe/) is two-dimensional size-mutable, potentially heterogeneous tabular data structure with labelled axes (rows and columns). A Data frame is a two-dimensional data structure, i.e., data is aligned in a tabular fashion in rows and columns. We can join, merge, and concat dataframe using different methods. In Dataframe df.merge(),df.join(), and df.concat() methods help in joining, merging and concating different



* **map = folium.Map(location=[20, 70], zoom\_start=4,tiles='Stamenterrain')**

[Folium](http://folium.readthedocs.io/en/latest/) is built on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js (JavaScript) library. Simply, manipulate your data in Python, then visualize it on a leaflet map via Folium. Folium makes it easy to visualize data that’s been manipulated in Python, on an interactive Leaflet map. This library has a number of built-in tilesets from OpenStreetMap, Mapbox etc.

for lat, lon, value, name in zip(df\_full['Latitude'], df\_full['Longitude'], df\_full['Confirmed'], df\_full['State/UnionTerritory']):

folium.CircleMarker([lat, lon], radius=value\*0.0015, popup = ('<strong>State</strong>: ' + str(name).capitalize() + '<br>''<strong>Total Cases</strong>: ' + str(value) + '<br>')color='red',fill\_color='red',fill\_opacity=0.3 ).add\_to(map).

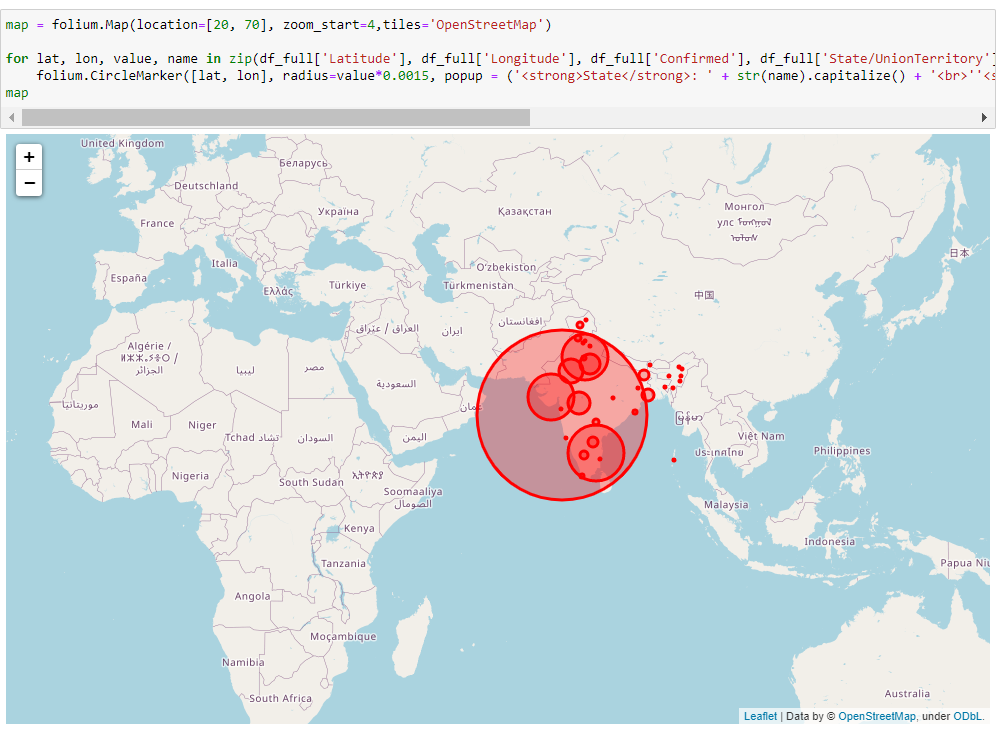
Step 1 . Create a Base Map.

Step 2 . A circular marker with popup text.

Step 3 . A simple\_marker for parachute style marker with pop-up text.

Step 4 . A line to the map

[Folium](http://folium.readthedocs.io/en/latest/) is built on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js (JavaScript) library. Simply, manipulate your data in Python, then visualize it on a leaflet map via Folium. Folium makes it easy to visualize data that’s been manipulated in Python, on an interactive Leaflet map. This library has a number of built-in tilesets from OpenStreetMap, Mapbox etc.



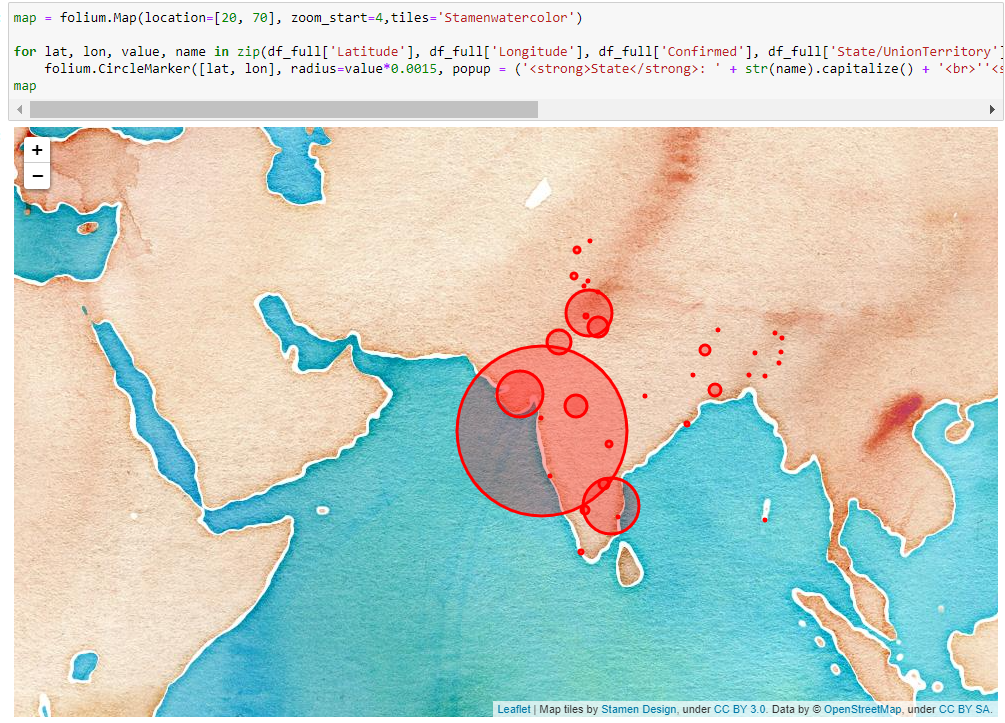
* map = folium.Map(location=[20, 70], zoom\_start=4,tiles='OpenStreetMap')

[Folium](http://folium.readthedocs.io/en/latest/) is built on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js (JavaScript) library. Simply, manipulate your data in Python, then visualize it on a leaflet map via Folium. Folium makes it easy to visualize data that’s been manipulated in Python, on an interactive Leaflet map. This library has a number of built-in tilesets from OpenStreetMap, Mapbox etc.

for lat, lon, value, name in zip(df\_full['Latitude'], df\_full['Longitude'], df\_full['Confirmed'], df\_full['State/UnionTerritory']):

folium.CircleMarker([lat, lon], radius=value\*0.0015, popup = ('<strong>State</strong>: ' + str(name).capitalize() + '<br>''<strong>Total Cases</strong>: ' + str(value) + '<br>')color='red',fill\_color='red',fill\_opacity=0.3 ).add\_to(map).

Here we are creating a map of title as OPENSTREETMAP



* map = folium.Map(location=[20, 70], zoom\_start=4,tiles='Stamenwatercolor')

[Folium](http://folium.readthedocs.io/en/latest/) is built on the data wrangling strengths of the Python ecosystem and the mapping strengths of the Leaflet.js (JavaScript) library. Simply, manipulate your data in Python, then visualize it on a leaflet map via Folium. Folium makes it easy to visualize data that’s been manipulated in Python, on an interactive Leaflet map. This library has a number of built-in tilesets from OpenStreetMap, Mapbox etc.

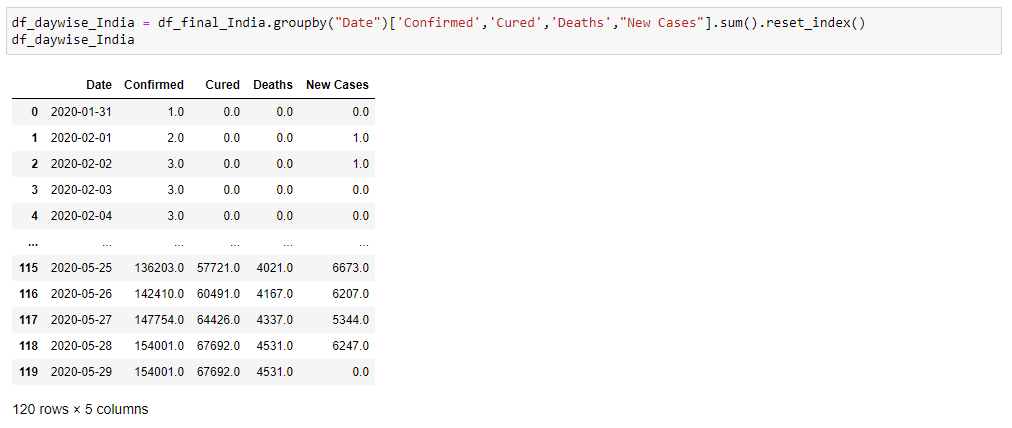
for lat, lon, value, name in zip(df\_full['Latitude'], df\_full['Longitude'], df\_full['Confirmed'], df\_full['State/UnionTerritory']):

folium.CircleMarker([lat, lon], radius=value\*0.0015, popup = ('<strong>State</strong>: ' + str(name).capitalize() + '<br>''<strong>Total Cases</strong>: ' + str(value) + '<br>'),color='red',fill\_color='red',fill\_opacity=0.3 ).add\_to(map)

map

Here we are creating a map of title as STAMEWATERCOLOR.

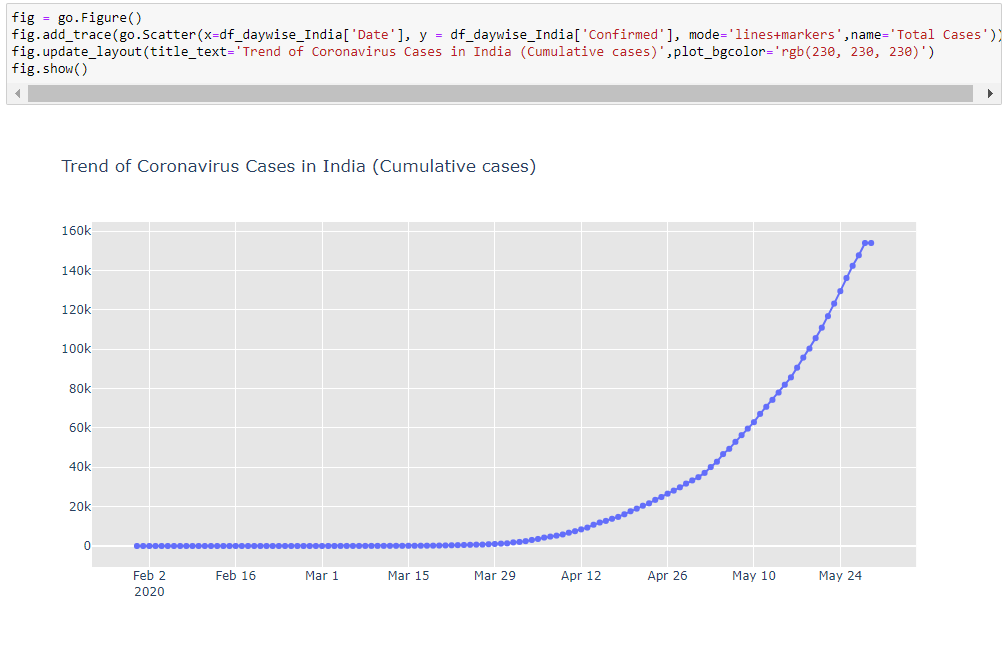
**TEND OF VIRUS**



Groupby() function is used with the sum(), and reset\_index() and a dataframe df\_daywise\_India Is made.

Pandas**dataframe.groupby()** function is used to split the data into groups based on some criteria. pandas objects can be split on any of their axes. The abstract definition of grouping is to provide a mapping of labels to group names. Pandas reset\_index() is a method to reset index of a Data Frame. reset\_index() method sets a list of integer ranging from 0 to length of data as index. Sum of numbers in the list is required everywhere. Python provide an inbuilt function sum() which sums up the numbers in the list.

Then we display the dataframe by calling out its name and the the first five rows and the last five rows are being displayed.



Now we create a object and then with the help of figure , scatter and Update\_layout we create a graph as shown above.

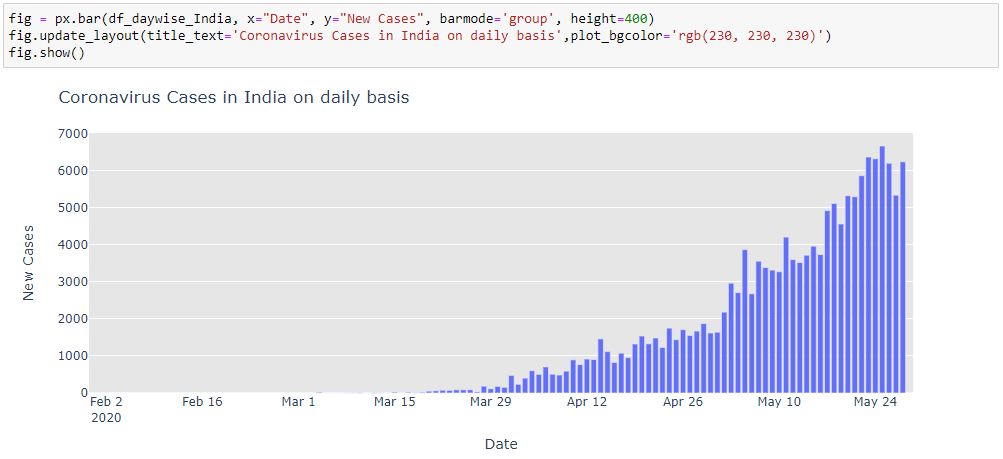
fig = go.Figure()

fig.add\_trace(go.Scatter(x=df\_daywise\_India['Date'], y = df\_daywise\_India['Confirmed'], mode='lines+markers',name='Total Cases'))

fig.update\_layout(title\_text='Trend of Coronavirus Cases in India (Cumulative cases)',plot\_bgcolor='rgb(230, 230, 230)')

fig.show()

and with the show() we display the graph.



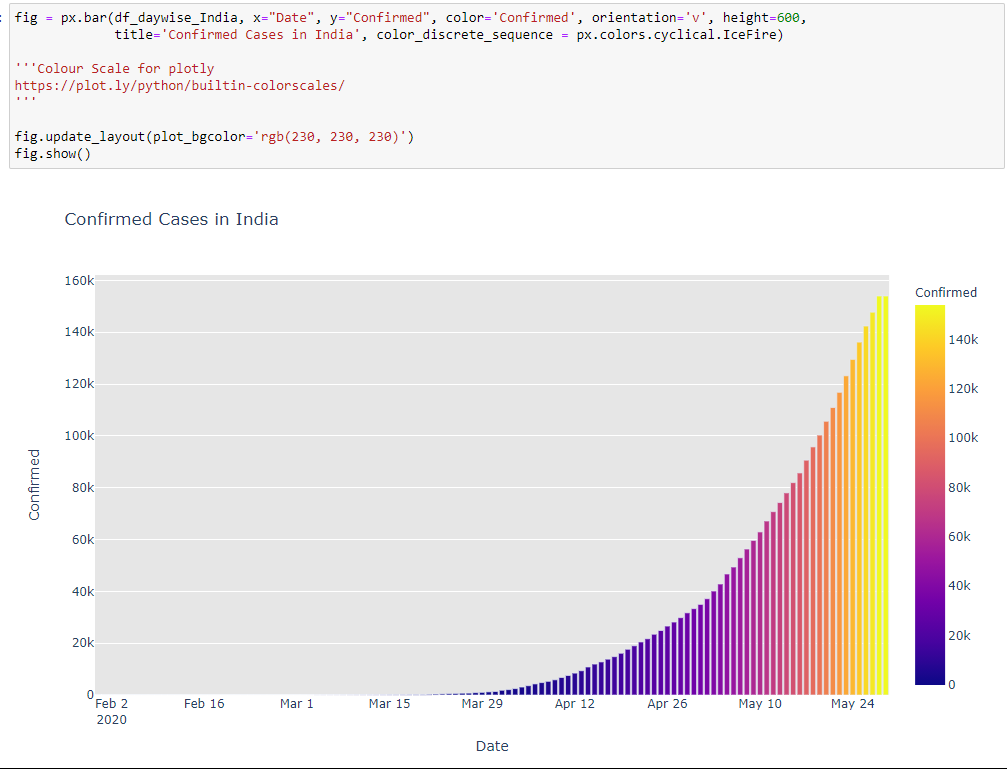
Now we create a object and then with the help of figure , bar and Update\_layout we create a graph as shown above.

ig = px.bar(df\_daywise\_India, x="Date", y="New Cases", barmode='group', height=400)

fig.update\_layout(title\_text='Coronavirus Cases in India on daily basis',plot\_bgcolor='rgb(230, 230, 230)')

fig.show()

and with the show() we display the graph.



Now we create a object and then with the help of figure , bar and Update\_layout we create a graph as shown above.

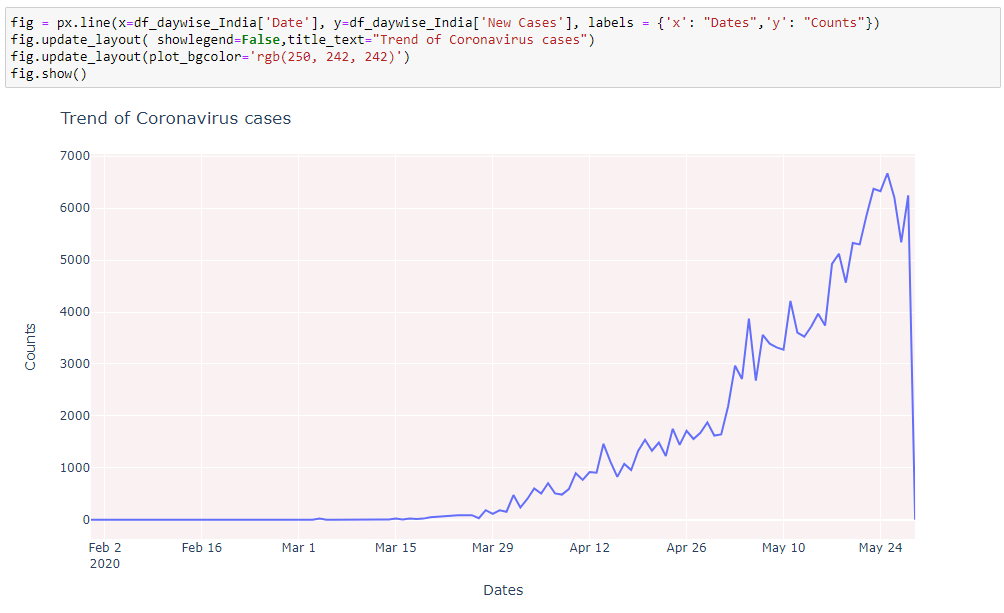
fig = px.bar(df\_daywise\_India, x="Date", y="Confirmed", color='Confirmed', orientation='v', height=600,

title='Confirmed Cases in India', color\_discrete\_sequence = px.colors.cyclical.IceFire)

fig.update\_layout(plot\_bgcolor='rgb(230, 230, 230)')

fig.show()

and with the show() we display the graph.



Now we create a object and then with the help of figure , line and Update\_layout we create a graph as shown above.

fig = px.line(x=df\_daywise\_India['Date'], y=df\_daywise\_India['New Cases'], labels = {'x': "Dates",'y': "Counts"})

fig.update\_layout( showlegend=False,title\_text="Trend of Coronavirus cases")

fig.update\_layout(plot\_bgcolor='rgb(250, 242, 242)')

fig.show()

and with the show() we display the graph.

**FBPROPHET**



First we need to import the Prophet from fbprophet,

Prophet follows the sklearn model API. We create an instance of the Prophet class and then call its fit and predict methods.

The input to Prophet is always a dataframe with two columns: ds and y. The ds (datestamp) column should be of a format expected by Pandas, ideally YYYY-MM-DD for a date or YYYY-MM-DD HH:MM:SS for a timestamp. The y column must be numeric, and represents the measurement we wish to forecast.

from fbprophet import Prophet;

df = df\_daywise\_India.iloc[:-1,]

df\_train = df.loc[df['Date']<= "2020-05-23",:]

df\_test = df.loc[df['Date'] > "2020-05-23",:]

here we select the column we need to act on so with the help of iloc we choose which column we need and which to omit.

confirmed\_train = df\_train[['Date','Confirmed']]

confirmed\_test = df\_test[['Date','Confirmed']]

deaths\_train = df\_train[['Date','Deaths']]

deaths\_test = df\_test[['Date','Deaths']]

recovered\_train = df\_train[['Date','Cured']]

recovered\_test = df\_test[['Date','Cured']]

Prepare separate training and testing model with the df\_train and df\_test of the confirmed , deaths and recovered .

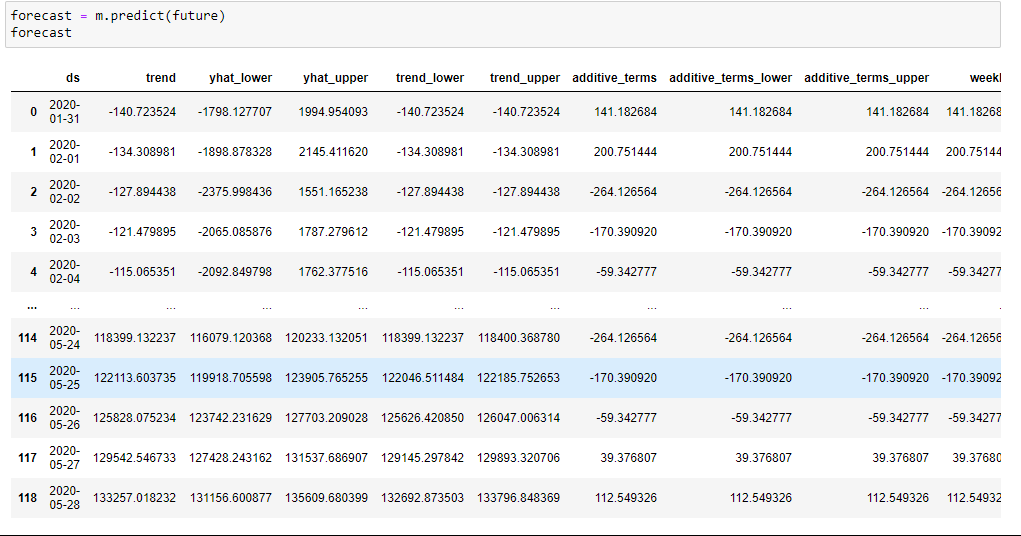


m = Prophet()

m.fit(confirmed\_train)

future = m.make\_future\_dataframe(periods=5,freq = "D")

future.tail(5)



With the help of predict we predict the future predictions.

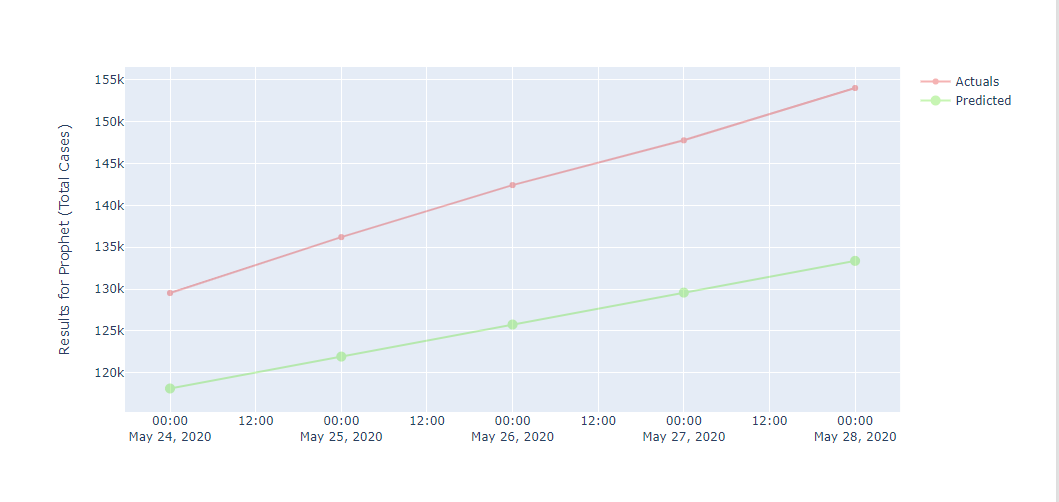
onfirmed\_train.columns = ['ds','y']

confirmed\_train.tail()



FBProphet uses time as a regressor and tries to fit several linear and nonlinear function of time as components. By default, FBProphet will fit the data using a linear model but it can be changed to the nonlinear model (logistics growth) from its arguments

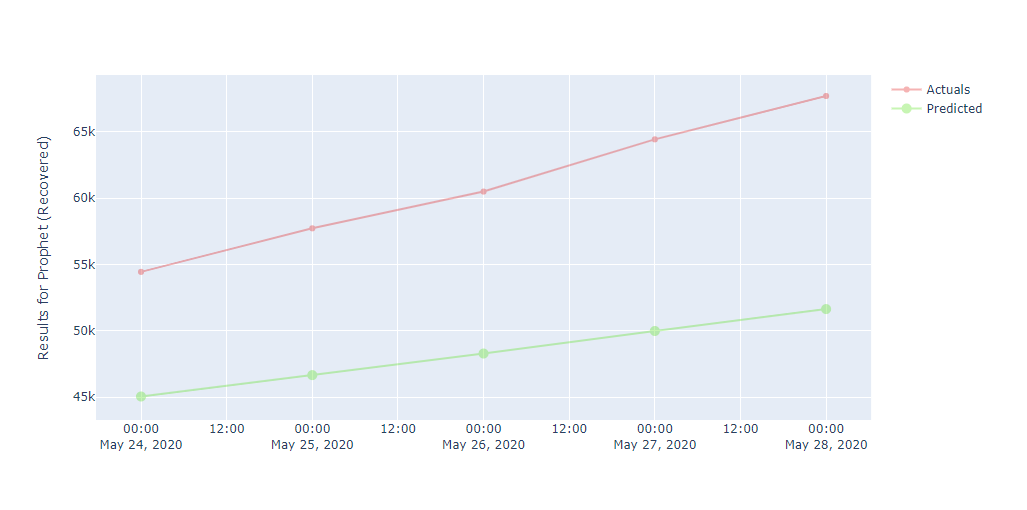
And hence we use forcast and after that with the scatter function of python we plot graph. As shown below,





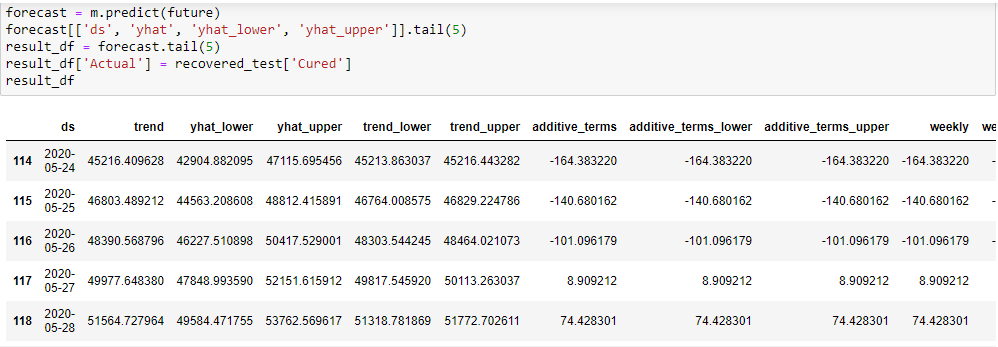
FBProphet uses time as a regressor and tries to fit several linear and nonlinear function of time as components. By default, FBProphet will fit the data using a linear model but it can be changed to the nonlinear model (logistics growth) from its arguments

And hence we use forcast and after that with the scatter function of python we plot graph. As shown below,



As we can see the above graph as the date move further the actual recovered are very more than our predicted one.





predict() : given a trained model, predict the label of a new set of data. This method accepts one argument, the new data X\_new (e.g. model. predict(X\_new) ), and returns the learned label for each object in the array.

forecast = m.predict(future)

forecast[['ds', 'yhat', 'yhat\_lower', 'yhat\_upper']].tail(5)

result\_df = forecast.tail(5)

result\_df['Actual'] = deaths\_test['Deaths']

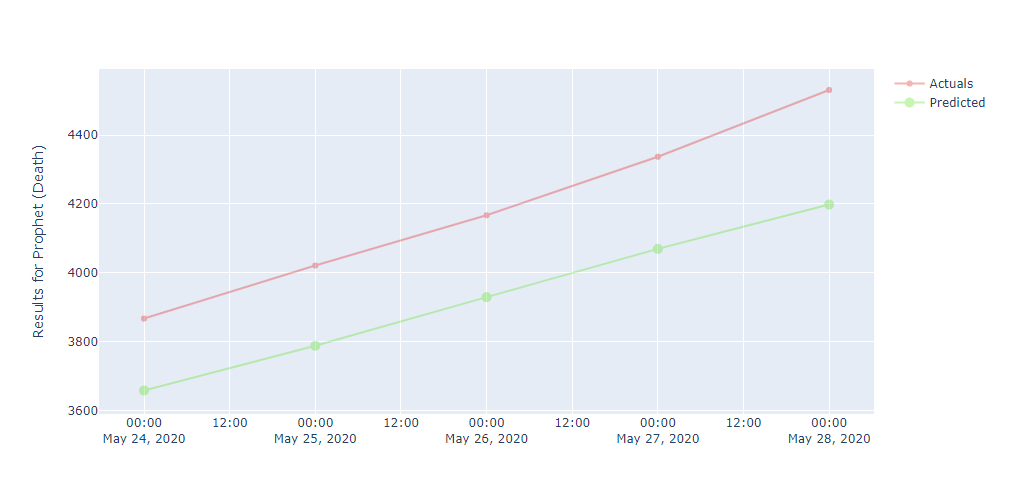
result\_df

hence we use predict and argument as future in it.



It uses time as a regressor and tries to fit several linear and nonlinear function of time as components. By default, FBProphet will fit the data using a linear model but it can be changed to the nonlinear model (logistics growth) from its arguments

And hence we use forcast and after that with the scatter function of python we plot graph. As shown below,



As we can see in the graph as we move further in dates results for actual deaths are more as compared to the predicted ones.

-------------------------------------------------------------------------------------------------------------

**COMPLETE CODE**

import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

import seaborn as sns

import plotly

import plotly.express as px

import plotly.graph\_objects as go

import folium

from folium import plugins

plt.rcParams['figure.figsize'] = 10, 12

import warnings

from sklearn.metrics import mean\_squared\_error

warnings.filterwarnings('ignore')

%matplotlib inline

df\_India= pd.read\_csv('covid\_19\_India.csv')

India\_coord = pd.read\_excel('Indian Coordinates.xlsx')

#guys i will send you these files and these are

#the file you have to use for your project

print(df\_India.info())

df\_India.head()

df\_India.tail()

df\_India.dtypes

print(India\_coord.info())

India\_coord.head()

def replace\_dash\_with\_zeros(inp):

return int(inp.replace("-","0"))

df\_India.drop(['Sno'],axis=1,inplace=True)

df\_India['Date'] = pd.to\_datetime(df\_India['Date'], format = "%d/%m/%y")

# https://www.stat.berkeley.edu/~s133/dates.html

df\_India['ConfirmedIndianNational'] = df\_India['ConfirmedIndianNational'].apply(replace\_dash\_with\_zeros)

df\_India['ConfirmedForeignNational'] = df\_India['ConfirmedForeignNational'].apply(replace\_dash\_with\_zeros)

df\_India.sort\_values("Confirmed", ascending = False, inplace = True)

df\_India

df\_India.loc[df\_India["ConfirmedForeignNational"] == "-",:]

list(zip(df\_India.columns,df\_India.dtypes,df\_India.isna().sum()))

print(f'We have data available from : {df\_India.Date.min()} to {df\_India.Date.max()}')

df\_India.groupby(["State/UnionTerritory", "Date"]).sum()

States = df\_India['State/UnionTerritory'].unique().tolist()

States

States.remove("Cases being reassigned to states")

States.remove("Unassigned")

States

len(States)

**MERGING THE DATAFRAMES**

df\_final\_India = pd.DataFrame()

dates = pd.DataFrame({"Date": pd.date\_range(df\_India.Date.min(),df\_India.Date.max())})

for state in States:

all\_dates\_df = pd.merge(dates,

df\_India.loc[df\_India['State/UnionTerritory'] == state,:], on = "Date",

how = "left")

all\_dates\_df['State/UnionTerritory'] = state

all\_dates\_df = all\_dates\_df.fillna(0)

all\_dates\_df['New Cases'] = all\_dates\_df['Confirmed'] - all\_dates\_df['Confirmed'].shift(1)

# print(state)

# display(all\_dates\_df.loc[all\_dates\_df['New Cases'] < 0,:])

df\_final\_India = pd.concat([df\_final\_India, all\_dates\_df],axis = 0)

print("Finally we have a data of Size: ",df\_final\_India.shape)

df\_final\_India.head()

df\_final\_India.dropna(inplace = True)

df\_final\_India.shape

del df\_final\_India['Time']

del df\_final\_India['ConfirmedIndianNational']

del df\_final\_India['ConfirmedForeignNational']

df\_final\_India

df\_final\_India.groupby(["State/UnionTerritory", "Date"]).sum()

df\_final\_India = df\_final\_India.groupby(["State/UnionTerritory", "Date"]).sum().reset\_index()

df\_final\_India

**STATEWISE COVID-19 STATUS**

def plot\_pie(active,cured,death,title):

labels = ['Active','Recovered','Died']

sizes = [active,cured,death]

color= ['#66b3ff','green','red']

explode = []

for i in labels:

explode.append(0.05)

plt.figure(figsize= (15,6))

plt.pie(sizes, labels=labels, autopct='%1.1f%%', startangle=9, explode =explode,colors = color)

centre\_circle = plt.Circle((0,0),0.70,fc='white')

fig = plt.gcf()

fig.gca().add\_artist(centre\_circle)

plt.title(title + 'COVID-19 Cases',fontsize = 20)

plt.axis('equal')

plt.tight\_layout()

total\_cases\_india = 0

cured\_cases\_india = 0

death\_cases\_india = 0

active\_cases\_india = 0

state\_df = pd.DataFrame()

for state in States:

one\_state\_df = df\_final\_India.loc[df\_final\_India['State/UnionTerritory'] == state,:]

state\_df = pd.concat([state\_df,pd.DataFrame(one\_state\_df.iloc[-1,:]).T],axis = 0)

total\_cases = one\_state\_df['Confirmed'].values[-1]

cured = one\_state\_df['Cured'].values[-1]

deaths = one\_state\_df['Deaths'].values[-1]

active = total\_cases - cured - deaths

plot\_pie(active, cured, deaths,state)

total\_cases\_india += total\_cases

cured\_cases\_india += cured

death\_cases\_india += deaths

active\_cases\_india += active

state\_df.reset\_index(inplace = True,drop = True)

state\_df

f, ax = plt.subplots(figsize=(12, 28))

data = state\_df[['State/UnionTerritory','Confirmed','Cured','Deaths']]

data.sort\_values('Confirmed',ascending=False,inplace=True)

sns.set\_color\_codes("pastel")

sns.barplot(x="Confirmed", y="State/UnionTerritory", data=data,label="Total", color="red")

sns.set\_color\_codes("muted")

sns.barplot(x="Cured", y="State/UnionTerritory", data=data, label="Cured", color="green")

ax.legend(ncol=5, loc="lower right", frameon=True)

ax.set(ylabel="",xlabel="Cases")

i = 0

for p in ax.patches:

x = p.get\_x() + p.get\_width() + 3

y = p.get\_y() + p.get\_height()/2

if i <= len(States):

ax.annotate(" "\*10 + str(int(p.get\_width())), (x, y))

else:

ax.annotate(int(p.get\_width()), (x, y))

i += 1

f, ax = plt.subplots(figsize=(12, 28))

data = state\_df[['State/UnionTerritory','Confirmed','Cured','Deaths']]

data.sort\_values('Confirmed',ascending=False,inplace=True)

sns.set\_color\_codes("pastel")

sns.barplot(x="Confirmed", y="State/UnionTerritory", data=data,label="Total", color="red")

sns.set\_color\_codes("muted")

sns.barplot(x="Cured", y="State/UnionTerritory", data=data, label="Cured", color="green")

ax.legend(ncol=5, loc="lower right", frameon=True)

ax.set(ylabel="",xlabel="Cases")

total = total\_cases\_india

i = 0

for p in ax.patches:

percentage = '{:.1f}%'.format(100 \* p.get\_width()/total)

x = p.get\_x() + p.get\_width() + 3

y = p.get\_y() + p.get\_height()/2

if i <= len(States):

ax.annotate(" "\*10 + str(percentage), (x, y))

else:

ax.annotate(percentage, (x, y))

i += 1

**OVERALL COVID-19 STATUS IN INDIA**

print("Total infected cases in India: ", total\_cases\_india)

print("Total cured cases in India: ", cured\_cases\_india)

print("Total active cases in India: ", active\_cases\_india)

print("Total death cases in India: ", death\_cases\_india)

plot\_pie(active\_cases\_india, cured\_cases\_india, death\_cases\_india, "India")

**VISUALISING THE SPREAD GEOGRAPHICALLY**

India\_coord.rename(columns = {"Name of State / UT" : "State/UnionTerritory"},inplace = True)

set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values))

India\_coord['State/UnionTerritory'] = India\_coord['State/UnionTerritory'].str.strip()

state\_df['State/UnionTerritory'] = state\_df['State/UnionTerritory'].str.strip()

set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values))

India\_coord.loc[India\_coord.shape[0]] = ['Gujarat','22.2587','71.1924']

India\_coord

set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values))

India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Andaman And Nicobar",

"Andaman and Nicobar Islands",India\_coord['State/UnionTerritory'])

India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Union Territory of Jammu and Kashmir",

"Jammu and Kashmir",India\_coord['State/UnionTerritory'])

India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Union Territory of Ladakh",

"Ladakh",India\_coord['State/UnionTerritory'])

India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Orissa",

"Odisha",India\_coord['State/UnionTerritory'])

India\_coord['State/UnionTerritory'] = np.where(India\_coord['State/UnionTerritory'] == "Dadra And Nagar Haveli",

"Dadar Nagar Haveli",India\_coord['State/UnionTerritory'])

set(India\_coord['State/UnionTerritory'].values).symmetric\_difference(set(state\_df['State/UnionTerritory'].values))

df\_full = pd.merge(India\_coord,state\_df,on='State/UnionTerritory').reset\_index(drop = True)

df\_full

map = folium.Map(location=[20, 70], zoom\_start=4,tiles='Stamenterrain')

for lat, lon, value, name in zip(df\_full['Latitude'], df\_full['Longitude'], df\_full['Confirmed'], df\_full['State/UnionTerritory']):

folium.CircleMarker([lat, lon], radius=value\*0.0015, popup = ('<strong>State</strong>: ' + str(name).capitalize() + '<br>''<strong>Total Cases</strong>: ' + str(value) + '<br>'),color='red',fill\_color='red',fill\_opacity=0.3 ).add\_to(map)

map

map = folium.Map(location=[20, 70], zoom\_start=4,tiles='OpenStreetMap')

for lat, lon, value, name in zip(df\_full['Latitude'], df\_full['Longitude'], df\_full['Confirmed'], df\_full['State/UnionTerritory']):

folium.CircleMarker([lat, lon], radius=value\*0.0015, popup = ('<strong>State</strong>: ' + str(name).capitalize() + '<br>''<strong>Total Cases</strong>: ' + str(value) + '<br>'),color='red',fill\_color='red',fill\_opacity=0.3 ).add\_to(map)

map

map = folium.Map(location=[20, 70], zoom\_start=4,tiles='Stamenwatercolor')

for lat, lon, value, name in zip(df\_full['Latitude'], df\_full['Longitude'], df\_full['Confirmed'], df\_full['State/UnionTerritory']):

folium.CircleMarker([lat, lon], radius=value\*0.0015, popup = ('<strong>State</strong>: ' + str(name).capitalize() + '<br>''<strong>Total Cases</strong>: ' + str(value) + '<br>'),color='red',fill\_color='red',fill\_opacity=0.3 ).add\_to(map)

map

**TEND OF VIRUS**

df\_daywise\_India = df\_final\_India.groupby("Date")['Confirmed','Cured','Deaths',"New Cases"].sum().reset\_index()

df\_daywise\_India

fig = go.Figure()

fig.add\_trace(go.Scatter(x=df\_daywise\_India['Date'], y = df\_daywise\_India['Confirmed'], mode='lines+markers',name='Total Cases'))

fig.update\_layout(title\_text='Trend of Coronavirus Cases in India (Cumulative cases)',plot\_bgcolor='rgb(230, 230, 230)')

fig.show()

fig = px.bar(df\_daywise\_India, x="Date", y="New Cases", barmode='group', height=400)

fig.update\_layout(title\_text='Coronavirus Cases in India on daily basis',plot\_bgcolor='rgb(230, 230, 230)')

fig.show()

fig = px.bar(df\_daywise\_India, x="Date", y="Confirmed", color='Confirmed', orientation='v', height=600,

title='Confirmed Cases in India', color\_discrete\_sequence = px.colors.cyclical.IceFire)

'''Colour Scale for plotly

https://plot.ly/python/builtin-colorscales/

'''

fig.update\_layout(plot\_bgcolor='rgb(230, 230, 230)')

fig.show()

fig = px.line(x=df\_daywise\_India['Date'], y=df\_daywise\_India['New Cases'], labels = {'x': "Dates",'y': "Counts"})

fig.update\_layout( showlegend=False,title\_text="Trend of Coronavirus cases")

fig.update\_layout(plot\_bgcolor='rgb(250, 242, 242)')

fig.show()

**FORECASTING USING FBPROPHET**

from fbprophet import Prophet

df = df\_daywise\_India.iloc[:-1,]

df\_train = df.loc[df['Date']<= "2020-05-23",:]

df\_test = df.loc[df['Date'] > "2020-05-23",:]

confirmed\_train = df\_train[['Date','Confirmed']]

confirmed\_test = df\_test[['Date','Confirmed']]

deaths\_train = df\_train[['Date','Deaths']]

deaths\_test = df\_test[['Date','Deaths']]

recovered\_train = df\_train[['Date','Cured']]

recovered\_test = df\_test[['Date','Cured']]

confirmed\_train.columns = ['ds','y']

confirmed\_train.tail()

m = Prophet()

m.fit(confirmed\_train)

future = m.make\_future\_dataframe(periods=5,freq = "D")

future.tail(5)

forecast = m.predict(future)

forecast

result\_df = forecast[['ds', 'yhat', 'yhat\_lower', 'yhat\_upper']].tail(5)

result\_df['Actual'] = confirmed\_test['Confirmed']

result\_df

trace0 = go.Scatter(

x = result\_df['ds'],

y = result\_df['Actual'],

mode = 'lines+markers',

name='Actuals',

line = dict(color = '#dd0000', shape = 'linear'),

opacity = 0.3,

connectgaps=True

)

trace1 = go.Scatter(

x = result\_df['ds'],

y = result\_df['yhat'],

name='Predicted',

mode = 'lines+markers',

marker = dict(

size = 10,

color = '#44dd00'),

opacity = 0.3

)

data = [trace0, trace1]

layout = go.Layout(

yaxis=dict(

title="Results for Prophet (Total Cases)"

)

)

fig = go.Figure(data=data, layout=layout)

fig.show()

recovered\_train.columns = ['ds','y']

recovered\_train.tail()

m = Prophet()

m.fit(recovered\_train)

future = m.make\_future\_dataframe(periods=5,freq = "D")

future.tail(5)

forecast = m.predict(future)

forecast[['ds', 'yhat', 'yhat\_lower', 'yhat\_upper']].tail(5)

result\_df = forecast.tail(5)

result\_df['Actual'] = recovered\_test['Cured']

result\_df

trace0 = go.Scatter(

x = result\_df['ds'],

y = result\_df['Actual'],

mode = 'lines+markers',

name='Actuals',

line = dict(color = '#dd0000', shape = 'linear'),

opacity = 0.3,

connectgaps=True

)

trace1 = go.Scatter(

x = result\_df['ds'],

y = result\_df['yhat'],

name='Predicted',

mode = 'lines+markers',

marker = dict(

size = 10,

color = '#44dd00'),

opacity = 0.3

)

data = [trace0, trace1]

layout = go.Layout(

yaxis=dict(

title="Results for Prophet (Recovered)"

)

)

fig = go.Figure(data=data, layout=layout)

fig.show()

deaths\_train.columns = ['ds','y']

deaths\_train.tail()

m = Prophet(seasonality\_mode= 'multiplicative')

m.fit(deaths\_train)

future = m.make\_future\_dataframe(periods=5,freq = "D")

future.tail(5)

forecast = m.predict(future)

forecast[['ds', 'yhat', 'yhat\_lower', 'yhat\_upper']].tail(5)

result\_df = forecast.tail(5)

result\_df['Actual'] = deaths\_test['Deaths']

result\_df

trace0 = go.Scatter(

x = result\_df['ds'],

y = result\_df['Actual'],

mode = 'lines+markers',

name='Actuals',

line = dict(color = '#dd0000', shape = 'linear'),

opacity = 0.3,

connectgaps=True

)

trace1 = go.Scatter(

x = result\_df['ds'],

y = result\_df['yhat'],

name='Predicted',

mode = 'lines+markers',

marker = dict(

size = 10,

color = '#44dd00'),

opacity = 0.3

)

data = [trace0, trace1]

layout = go.Layout(

yaxis=dict(

title="Results for Prophet (Death)"

)

)

fig = go.Figure(data=data, layout=layout)

fig.show()

=================================================================

**CONCLUSTION**

* In our project, we import two files

1. covid 19 india.csv
2. Indian Coordinates.xls

* We then prepare our data for analysis with th help of data cleaning by removing or modifying data that is incorrect, incomplete, irrelevant, duplicated, or improperly formatted.
* Then we perform the merging operation wherein we join the two dataframes. In this way we get all the information about a particular entity commin to both the two datasets.
* Each state's cases are analysed using a pie chart. Also, overall country's cases are analysed.

1. States like Maharashtra, Gujarat, MP West Bengal have death rate nearly 5 percent. ii. States like Tamil nadu, MP, UP, Rajasthan, Andrapradesh,
2. Telangana, Haryana, Kerela have recovered more than 50 percent.
3. Punjab has almost 90 percent cases recovered. iv. No death cases have been recorded in Tripura, Goa, Ladakh.
4. Pondicherry and Manipura. Andaman and Nicobar Islands have all the cases recovered. vi. Dadra and Nagar Haveli, Sikkim, Nagaland has all the cases active.
5. Through this project, the analysis on COVID-19 data has been performed successfully. The
6. analysis on this pandemic spread has been done and compared between different countries. The
7. analysis of confirmed cases, active cases, recovered cases and deaths are done separately to
8. give a clear look on how the virus is spreading, which countries are getting affected mostly and
9. how different countries are recovering. A separate analysis on cases of INDIA has been done
10. and predictions of different cases both around the world and INDIA has been done. At last, the
11. accuracy check using different metrics is performed over all the analysis done in this project
12. Through this project, the analysis on COVID-19 data has been performed successfully. The
13. analysis on this pandemic spread has been done and compared between different countries. The
14. analysis of confirmed cases, active cases, recovered cases and deaths are done separately to
15. give a clear look on how the virus is spreading, which countries are getting affected mostly and
16. how different countries are recovering. A separate analysis on cases of INDIA has been done
17. and predictions of different cases both around the world and INDIA has been done. At last, the
18. accuracy check using different metrics is performed over all the analysis done in this project

Through this project, the analysis on COVID-19 data has been performed successfully. The analysis on this pandemic spread has been done and compared between different countries. The analysis of confirmed cases, active cases, recovered cases and deaths are done separately to give a clear look on how the virus is spreading, which countries are getting affected mostly and how different countries are recovering. A separate analysis on cases of INDIA has been done and predictions of different cases both around the world and INDIA has been done. At last, the accuracy check using different metrics is performed over all the analysis done in this project.