Dhirubhai Ambani
Institute of Information and Communication Technology

Covid-19 Analysis

Project Report

Subject: Python

Subject code: IT606

Name:

Tanya Jagyasi (202118039)

Bhumi Bosamia (202118040)

Problem Statement :

The patient details from a dataset can't be found directly, such as knowing the growth rate or comparing the cases between the state, district, or nation regarding the active, confirmed, recovered, and deaths cases. The aim is to show the Exploratory Analysis to get insights of covid patients, covid growth on the state, district, national level.

INTRODUCTION :

COVID-19 outbreak was first reported in Wuhan, China, and has spread to more than 50 countries. Since the COVID-19 outbreak in China, the first coronavirus case in India, the second most populated country globally, was reported in Kerala. New confirmed cases were reported in cities such as New Delhi, Mumbai, Bengaluru, Hyderabad, and Patna. COVID-19 is still an unclear infectious disease. We have a lot of datasets on the patients affected by Covid-19. Here we have taken the dataset for India.

So, based on the reports, we have done an analysis.

Covid-19 Analysis

Starting with importing the libraries

```
import numpy as np
```

Numpy: stands for Numerical Python, a library consisting of multidimensional array objects and a collection of routines for processing those arrays. Using NumPy, mathematical and logical operations on arrays can be performed. NumPy is a Python package.

```
import pandas as pd
```

Pandas: Pandas is a predominantly used python data analysis library. It provides many functions and methods to expedite the data analysis process.

```
import matplotlib.pyplot as plt
```

```
from matplotlib.pyplot import pie,axis,show
```

Matplotlib: Matplotlib is a cross-platform, data visualization and graphical plotting library for Python, and its numerical extension is NumPy.

```
import plotly.offline as pyo
```

```
from plotly.offline import iplot
```

Ploty: Plotly's Python graphing library makes interactive, publication-quality graphs. Examples of how to make line plots, scatter plots, area charts, bar charts, error bars, box plots, histograms, heatmaps, subplots, multiple-axes, polar charts, and bubble charts.

```
import cufflinks as cf
```

```
cf.go_offline()
```

Cufflinks: Library that connects the Pandas data frame with Plotly enabling users to create visualizations directly from Pandas. The library binds the power of Plotly with the flexibility of Pandas for easy plotting.

Covid-19 Analysis

```
import plotly.io as pio

pio.renderers.default='colab'
```

Plotly.io: Low-level interface for displaying, reading, and writing figures.

```
import seaborn as sns
```

Seaborn: An open-source Python library built on top of matplotlib. It is used for data visualization and exploratory data analysis. Seaborn works efficiently with data frames and the Pandas library. The graphs created can also be customized easily.

We have imported from Google drive using the command:
`drive.mount('/content/gdrive')`

and then we have read four CSV files:

covid_data - contains the longitude and latitude of the state

district_data - contains information about the district

national_data - contains information about nation

state_data - contains information about state

Covid-19 Analysis

Data Cleaning

Moving Forward to Data Cleaning part:

First, we have done some cleaning on district dataset

- We have renamed our columns

  using command:

```
district_data.rename(columns={'Deceased':'Deaths'},inplace=True)
```

- We Summed up the null values of all the columns

  using command:

```
district_data.isna().sum()
```

- We dropped the unwanted columns

  using command:

```
district_data =
district_data.drop(['District_Notes','Last_Updated'],axis
=1)
```

- We displayed the information about the district data set:

  Using `district_data.info()` command.

  It gives an overview of the dataset, such as displaying the total number of rows and columns; for example, it has 800 rows and 14 columns. It also shows the data type of each column of our dataset and represents the total count of not null values of each column. It is generally used in exploratory data analysis.

Covid-19 Analysis

- We checked all the negative values in the columns

  using command:

```
check = {}
for columns in district_data:
 if district_data[columns].dtypes == "int64":
   s = district_data[columns].lt(0).sum()
   check[columns]=s
print(check)
```

- Convert the negative values to absolute value

  code:

```
for key in check:
 if check[key] != 0:
   district_data[key] = district_data[key].abs()
```

- Verify it again

  code:

```
verify={}
for columns in district_data:
 if district_data[columns].dtypes == "int64":
   s = district_data[columns].lt(0).sum()
   verify[columns]=s
print(verify)
```

Covid-19 Analysis

- describe the data

```
district_data.describe()
```

Similarly, we have cleaned all the other datasets like:

Covid Data :

Complete Covid Data

```
covid_data
```

	Date	Name of State / UT	Latitude	Longitude	Total Confirmed cases	Death	Cured/Discharged/Migrated	New cases	New deaths	New recovered
0	2020-01-30	Kerala	10.8505	76.2711	1.0	0	0.0	0	0	0
1	2020-01-31	Kerala	10.8505	76.2711	1.0	0	0.0	0	0	0
2	2020-02-01	Kerala	10.8505	76.2711	2.0	0	0.0	1	0	0
3	2020-02-02	Kerala	10.8505	76.2711	3.0	0	0.0	1	0	0
4	2020-02-03	Kerala	10.8505	76.2711	3.0	0	0.0	0	0	0
...	...	...	...	...	...	...	...	...	...	...
4687	2020-08-06	Telangana	18.1124	79.0193	73050.0	589	52103.0	2092	0	1289
4688	2020-08-06	Tripura	23.9408	91.9882	5725.0	31	3793.0	97	0	68
4689	2020-08-06	Uttar Pradesh	26.8467	80.9462	104388.0	1857	60558.0	4078	0	3287
4690	2020-08-06	Uttarakhand	30.0668	79.0193	8254.0	98	5233.0	246	0	386
4691	2020-08-06	West Bengal	22.9868	87.8550	83800.0	1846	58962.0	2816	0	2078

4692 rows × 10 columns

```
[ ] # To count the missing Values
    covid_data.isna().sum()
```

```
Date                      0
Name of State / UT        0
Latitude                  0
Longitude                 0
Total Confirmed cases     0
Death                     0
Cured/Discharged/Migrated 0
New cases                 0
New deaths                0
New recovered             0
dtype: int64
```

Covid-19 Analysis

```
[ ] # To make all negative entries positive
    check={}
    for columns in covid_data:
      if covid_data[columns].dtypes != "object":
        s = covid_data[columns].lt(0).sum()
        check[columns]=s
    print(check)
```

```
{'Latitude': 0, 'Longitude': 0, 'Total Confirmed cases': 0, 'Cured/Discharged/Migrated': 0, 'New cases': 0, 'New deaths': 0, 'New recovered': 3}
```

```
#To convert negative values to abs:
for key in check:
  if check[key] != 0:
    covid_data[key] = covid_data[key].abs()
```

+ Code + Text

```
[ ] # To verify whether values are still negative or not:
    verify={}
    for columns in covid_data:
      if covid_data[columns].dtypes != "object":
        s = covid_data[columns].lt(0).sum()
        verify[columns]=s
    print(verify)
```

```
{'Latitude': 0, 'Longitude': 0, 'Total Confirmed cases': 0, 'Cured/Discharged/Migrated': 0, 'New cases': 0, 'New deaths': 0, 'New recovered': 0}
```

```
[ ] covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4692 entries, 0 to 4691
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Date                      4692 non-null   object
 1   Name of State / UT        4692 non-null   object
 2   Latitude                  4692 non-null   float64
 3   Longitude                 4692 non-null   float64
 4   Total Confirmed cases     4692 non-null   float64
 5   Death                     4692 non-null   object
 6   Cured/Discharged/Migrated 4692 non-null   float64
 7   New cases                 4692 non-null   int64
 8   New deaths                4692 non-null   int64
```

```
[ ]     s = covid_data[columns].lt(0).sum()
        verify[columns]=s
    print(verify)
```

```
{'Latitude': 0, 'Longitude': 0, 'Total Confirmed cases': 0, 'Cured/Discharged/Migrated': 0, 'New cases': 0, 'New deaths': 0, 'New recovered': 0}
```

```
covid_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4692 entries, 0 to 4691
Data columns (total 10 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   Date                      4692 non-null   object
 1   Name of State / UT        4692 non-null   object
 2   Latitude                  4692 non-null   float64
 3   Longitude                 4692 non-null   float64
 4   Total Confirmed cases     4692 non-null   float64
 5   Death                     4692 non-null   object
 6   Cured/Discharged/Migrated 4692 non-null   float64
 7   New cases                 4692 non-null   int64
 8   New deaths                4692 non-null   int64
 9   New recovered             4692 non-null   int64
dtypes: float64(4), int64(3), object(3)
memory usage: 366.7+ KB
```

```
[ ] covid_data.describe()
```

	Latitude	Longitude	Total Confirmed cases	Cured/Discharged/Migrated	New cases	New deaths	New recovered
count	4692.000000	4692.000000	4692.000000	4692.000000	4692.000000	4692.0	4692.000000
mean	23.185327	81.451837	11393.925192	6908.130648	418.643009	0.0	283.070332
std	6.635913	6.959475	37208.600846	23390.671258	1259.748923	0.0	947.925429
min	0.000000	0.000000	1.000000	0.000000	0.000000	0.0	0.000000
25%	18.112400	76.271100	39.000000	9.000000	1.000000	0.0	0.000000
50%	23.940800	79.019300	619.000000	197.500000	26.000000	0.0	8.000000
75%	28.218000	85.313100	5233.000000	2736.000000	210.250000	0.0	119.000000
max	34.299600	94.727800	468265.000000	305521.000000	18366.000000	0.0	13401.000000

Covid-19 Analysis

National Data:

National Level Data

```
[ ] # To remove the spaces in the column names:
    national_data.columns = national_data.columns.str.replace(" ","")
```

```
[ ] national_data.columns
```

```
Index(['Date', 'DailyConfirmed', 'TotalConfirmed', 'DailyRecovered',
       'TotalRecovered', 'DailyDeceased', 'TotalDeceased'],
      dtype='object')
```

```
[ ] national_data.rename(columns={'TotalDeceased':'TotalDeaths'},inplace=True)
```

```
national_data.isna().sum()
```

```
Date              0
DailyConfirmed    0
TotalConfirmed    0
DailyRecovered    0
TotalRecovered    0
DailyDeceased     0
TotalDeaths       0
dtype: int64
```

```
[ ] national_data.dtypes
```

```
Date              object
DailyConfirmed     int64
TotalConfirmed     int64
DailyRecovered     int64
TotalRecovered     int64
DailyDeceased      int64
TotalDeaths        int64
dtype: object
```

```
[ ] check = {}
    for columns in national_data:
      if national_data[columns].dtypes == "int64":
        s = national_data[columns].lt(0).sum()
        check[columns]=s
    print(check)
```

```
{'DailyConfirmed': 0, 'TotalConfirmed': 0, 'DailyRecovered': 0, 'TotalRecovered': 0, 'DailyDeceased': 0, 'TotalDeaths': 0}
```

```
national_data.describe()
```

	DailyConfirmed	TotalConfirmed	DailyRecovered	TotalRecovered	DailyDeceased	TotalDeaths
count	190.000000	1.900000e+02	190.000000	1.900000e+02	190.000000	190.000000
mean	10660.121053	3.088709e+05	7248.815789	1.869665e+05	219.226316	7849.294737
std	15739.372014	4.939299e+05	11755.421065	3.223640e+05	291.382213	11371.176250
min	0.000000	1.000000e+00	0.000000	0.000000e+00	0.000000	0.000000
25%	22.750000	1.522500e+02	1.000000	1.500000e+01	0.250000	3.000000
50%	2961.500000	4.460600e+04	1022.000000	1.230400e+04	91.500000	1514.500000
75%	14512.250000	4.077742e+05	10594.750000	2.246905e+05	383.250000	13201.000000
max	62170.000000	2.025423e+06	51368.000000	1.377275e+06	2004.000000	41653.000000

Covid-19 Analysis

State Data:

State Level Latest data

```
[ ] state_data
```

	State	Confirmed	Recovered	Deaths	Active	Last_Updated_Time	Migrated_Other	State_code	Delta_Confirmed	Delta_Recovered	Delta_Deaths	State_Notes
0	Total	2025409	1377384	41638	605933	06/08/2020 23:46:37	454	TT	0	0	0	NaN
1	Maharashtra	479779	316375	16792	146305	06/08/2020 20:42:51	307	MH	0	0	0	307 cases are marked as non-covid deaths in MH...
2	Tamil Nadu	279144	221087	4571	53486	06/08/2020 19:44:47	0	TN	0	0	0	[July 22]: 444 backdated deceased entries adde...
3	Delhi	141531	127124	4059	10348	06/08/2020 18:39:45	0	DL	0	0	0	[July 14]: Value for the total tests conducted...
4	Karnataka	158254	80281	2897	75067	06/08/2020 21:19:51	9	KA	0	0	0	NaN
5	Andhra Pradesh	196789	112870	1753	82166	06/08/2020 19:45:03	0	AP	0	0	0	Total includes patients from other states and ...
6	Uttar Pradesh	108974	63402	1918	43654	06/08/2020 17:53:49	0	UP	0	0	0	NaN
7	Gujarat	67811	50524	2579	14708	06/08/2020 21:28:40	0	GJ	0	0	0	NaN
8	West Bengal	86754	61023	1902	23829	06/08/2020 21:28:42	0	WB	0	0	0	NaN
9	Telangana	73050	52103	589	20358	06/08/2020 11:26:06	0	TG	0	0	0	[July 27]\nTelangana bulletin for the previous...
10	Rajasthan	48996	35131	757	13108	06/08/2020 21:54:44	0	RJ	0	0	0	NaN
11	Bihar	68148	43820	388	23939	06/08/2020 17:29:43	1	BR	0	0	0	NaN
12	Haryana	39303	32640	458	6205	06/08/2020 20:29:05	0	HR	0	0	0	[Aug 2]: 21 Foreign Evacuees have been merged ...
13	Assam	52818	37225	126	15464	06/08/2020 23:00:49	3	AS	0	0	0	Includes 1 case from Nagaland
14	Madhya Pradesh	36564	26902	946	8716	06/08/2020 19:55:50	0	MP	0	0	0	NaN
15	Odisha	40717	26888	280	13549	06/08/2020 20:44:03	0	OR	0	0	0	[July 12th] 20 non-covid deaths reported in st...
16	Jammu and Kashmir	23454	15708	436	7310	06/08/2020 18:19:43	0	JK	0	0	0	NaN
17	Kerala	30449	18333	98	11983	06/08/2020 20:08:58	35	KL	0	0	0	Mahe native who expired in Kannur included in ...
18	Punjab	20891	13659	517	6715	06/08/2020 21:19:59	0	PB	0	0	0	NaN
19	Jharkhand	15756	6594	145	9017	06/08/2020 23:00:52	0	JH	0	0	0	NaN
20	Chhattisgarh	11020	8088	77	2855	06/08/2020 23:00:55	0	CT	0	0	0	NaN
21	Uttarakhand	8623	5427	102	3056	06/08/2020 23:46:40	38	UT	0	0	0	NaN

```
state_data.isna().sum()
```

```
State                0
Confirmed            0
Recovered            0
Deaths               0
Active               0
Last_Updated_Time    0
Migrated_Other       0
State_code           0
Delta_Confirmed      0
Delta_Recovered      0
Delta_Deaths         0
State_Notes         23
dtype: int64
```

```
[ ] state_data = state_data.drop(['State_Notes'],axis=1)
```

```
[ ] check ={}
    for columns in state_data:
      if state_data[columns].dtypes == "int64":
        s = state_data[columns].lt(0).sum()
        check[columns]=s
    print(check)
```

```
{'Confirmed': 0, 'Recovered': 0, 'Deaths': 0, 'Active': 0, 'Migrated_Other': 0, 'Delta_Confirmed': 0, 'Delta_Recovered': 0, 'Delta_Deaths': 0}
```

```
[ ] state_data.describe()
```

	Confirmed	Recovered	Deaths	Active	Migrated_Other	Delta_Confirmed	Delta_Recovered	Delta_Deaths
count	3.800000e+01	3.800000e+01	38.000000	38.000000	38.000000	38.0	38.0	38.0
mean	1.066005e+05	7.249389e+04	2191.473684	31891.210526	23.894737	0.0	0.0	0.0
std	3.332312e+05	2.267774e+05	7163.656211	100001.749436	87.402081	0.0	0.0	0.0
min	0.000000e+00	0.000000e+00	0.000000	0.000000	0.000000	0.0	0.0	0.0
25%	2.106000e+03	1.194000e+03	9.250000	849.250000	0.000000	0.0	0.0	0.0
50%	1.832350e+04	1.087350e+04	114.000000	6460.000000	0.000000	0.0	0.0	0.0
75%	6.806375e+04	4.884800e+04	898.750000	15275.000000	2.500000	0.0	0.0	0.0

Covid-19 Analysis

DATA ANALYSIS

Starting with the analysis part:

Firstly, we extracted the date on which the nation had the highest number of recovered cases.

```
national_data ['Date'][national_data.TotalRecovered ==
national_data. TotalRecovered.max()]
```

Then using the District Data, we have extracted the name of the state which has the highest number of confirmed cases and the details of that State.

```
sc = district_data ['State'][district_data .Confirmed == district_data
.Confirmed.max()]
```

```
data = district_data .loc[district_data ['Confirmed'].idxmax()]

print("Detailed Information about the state having highest confirmed
cases \n",data)
```

Next, we have created a new variable containing the top 35 places with the highest number of confirmed cases.

```
top35 = district_data .nlargest(35,'Confirmed')
```

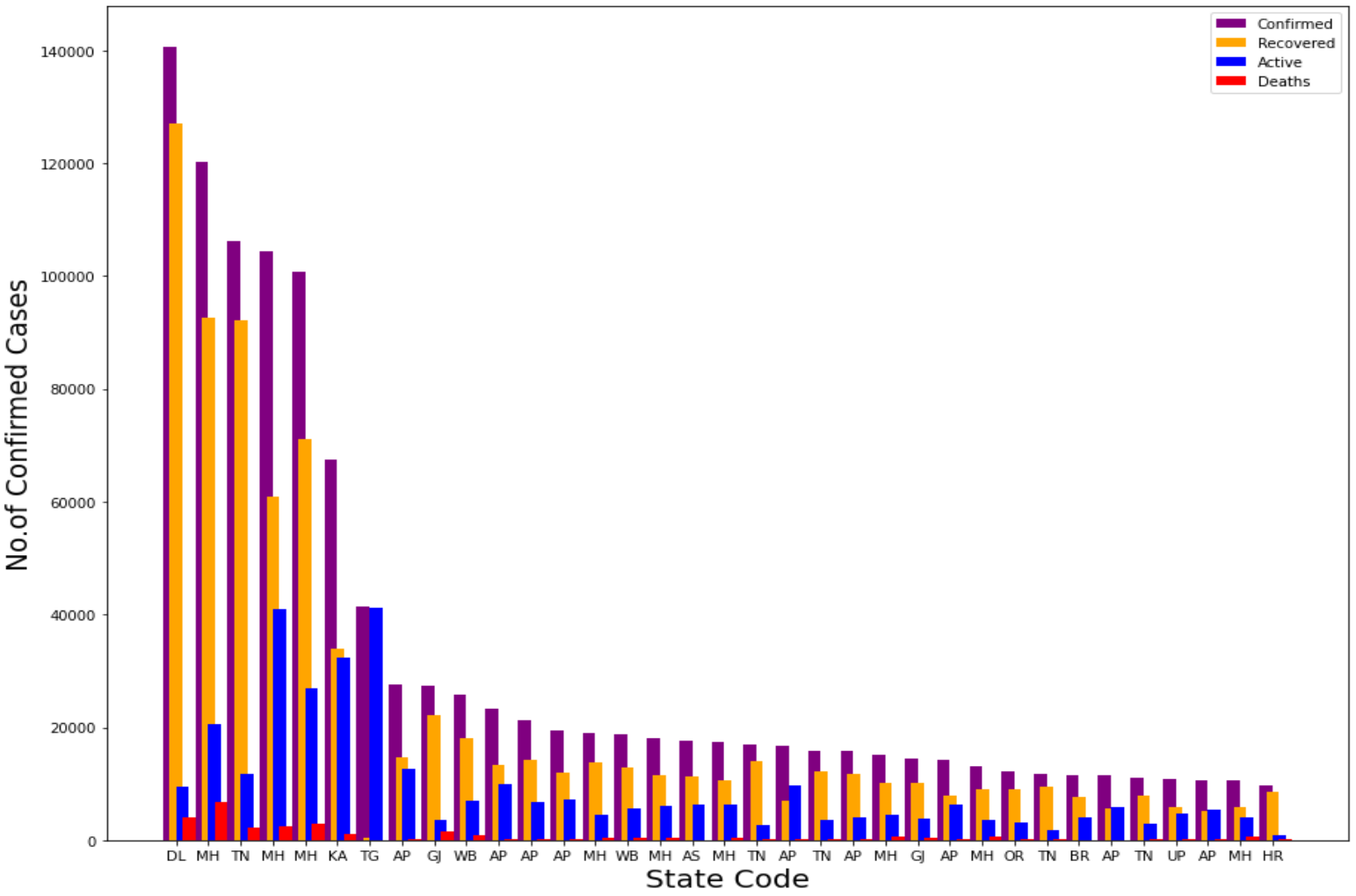
Covid-19 Analysis

Bar Plot :

Now using the top 35 data, we have plotted the bar graph, which displays the Confirmed, Recovered, Active, and Death cases according to the State Code. Here we can see of each State Code the highest number of the particular cases and how it varies as we move from left to right.

```
ind = np.arange(35)

width = 0.4



plt.figure(figsize=(15,12))

x = top35['State_Code']

y = top35['Confirmed']

plt.bar(ind+width/2,y,align='edge',width=width,label="Confirmed",
color='Purple')

y = top35['Recovered']

plt.bar(ind+width,y,align='edge',width=width,label="Recovered",
color='orange')

y = top35['Active']

plt.bar(ind+3*width/2,y,align='edge',width=width,label="Active",color='
Blue')

y = top35['Deaths']

plt.bar(ind+2*width,y,align='edge',width=width,label="Deaths",
color='Red')



plt.xticks(ind + 3*width/2, x)

plt.xlabel('State Code',fontsize=20)

plt.ylabel('No.of Confirmed Cases',fontsize=20)

plt.legend();
```

Covid-19 Analysis

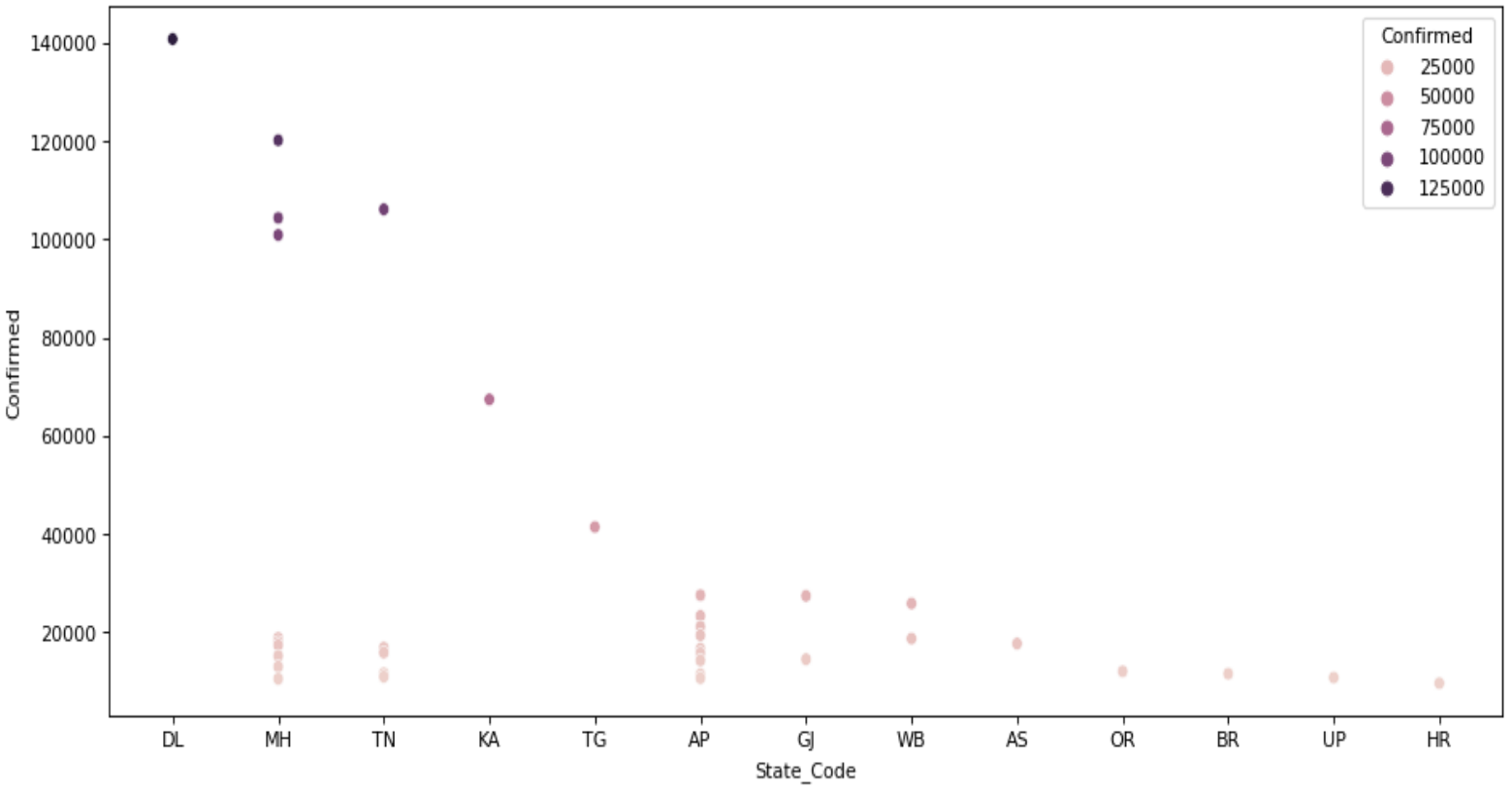


Scatter Plots :

Here we have a scatter plot that displays how the confirmed cases are spread in each state. Here the dark dots represent a higher no. of cases, and the light dots represent a lower no. of cases. Here we can see that states like Delhi, Maharashtra, and Tamil Nadu have more cases than others. In Maharashtra, we can say it had fewer cases in some time frame while it had more for a certain period of time.

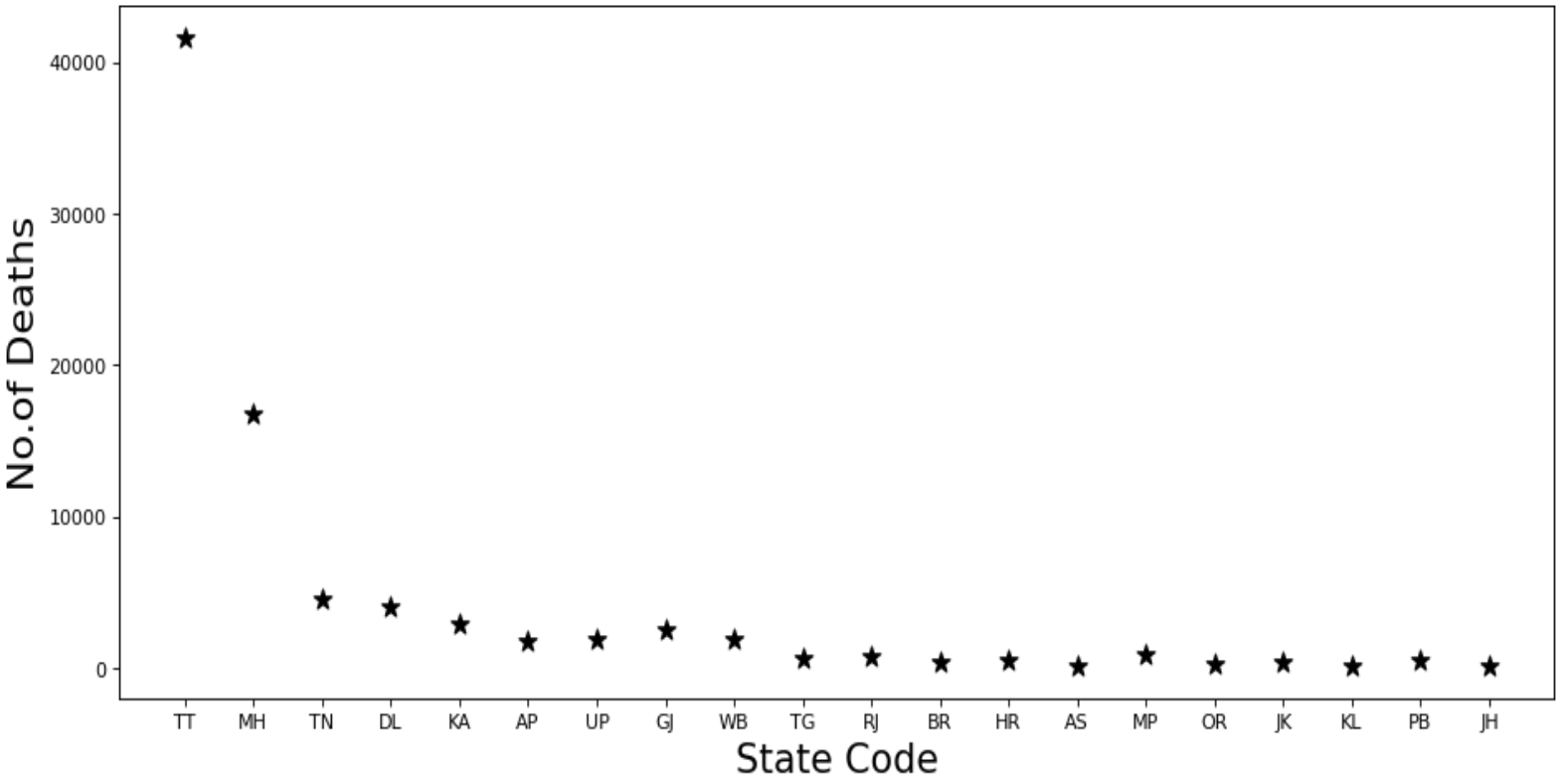
```
sc = sns.scatterplot(x='State_Code', y ='Confirmed',data = top35, hue =
'Confirmed')
```

Covid-19 Analysis



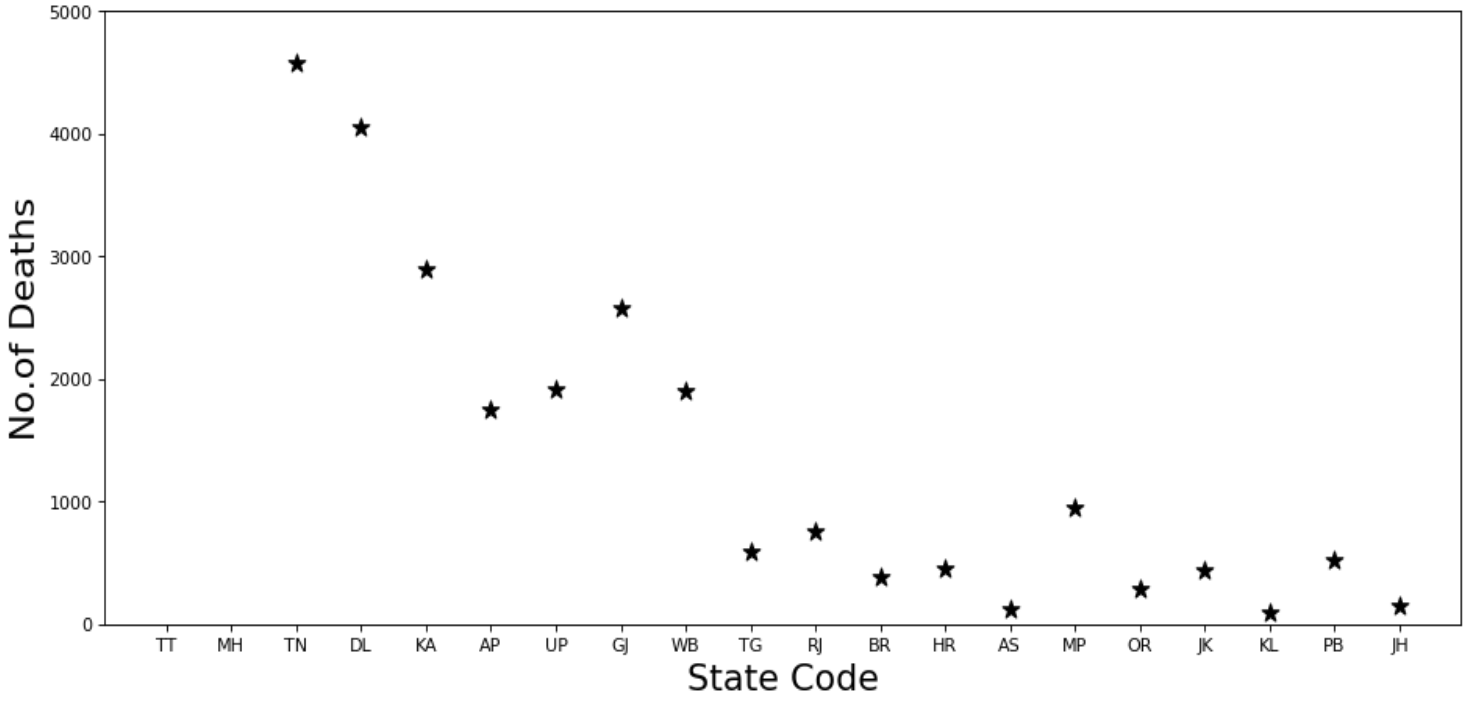
In the next scatter plot, we can see how the death cases are spread. Here TT represents the total, and we can see it is more than 40000 and here, Maharashtra has the highest no. of death cases, and as we move to Jharkhand, it decreases. It shows that states between Tamil Nadu and Jharkhand have less than 10000 cases but a clear picture is not seen.

Covid-19 Analysis



So using the following graph, we can see that there are states with death cases less than 1000 also.

Covid-19 Analysis



HeatMap :

Here we have a heat-map that shows the correlation between each column. As we know, the correlation of anything with itself is one; so we can see that the diagonal values are one which saws the correlation of each column with itself. The value of the correlation between Confirmed and Recovered is 0.946, so we can say they are highly correlated i.e., as the number of Confirmed cases increases, the number of Recovered cases is also increasing.

Similarly, the value between Migrated Other and Delta Active is 0.024, so we can say that they are the least related. If we had any negative value, we could say that they are negatively correlated i.e as one increases, the other decreases.

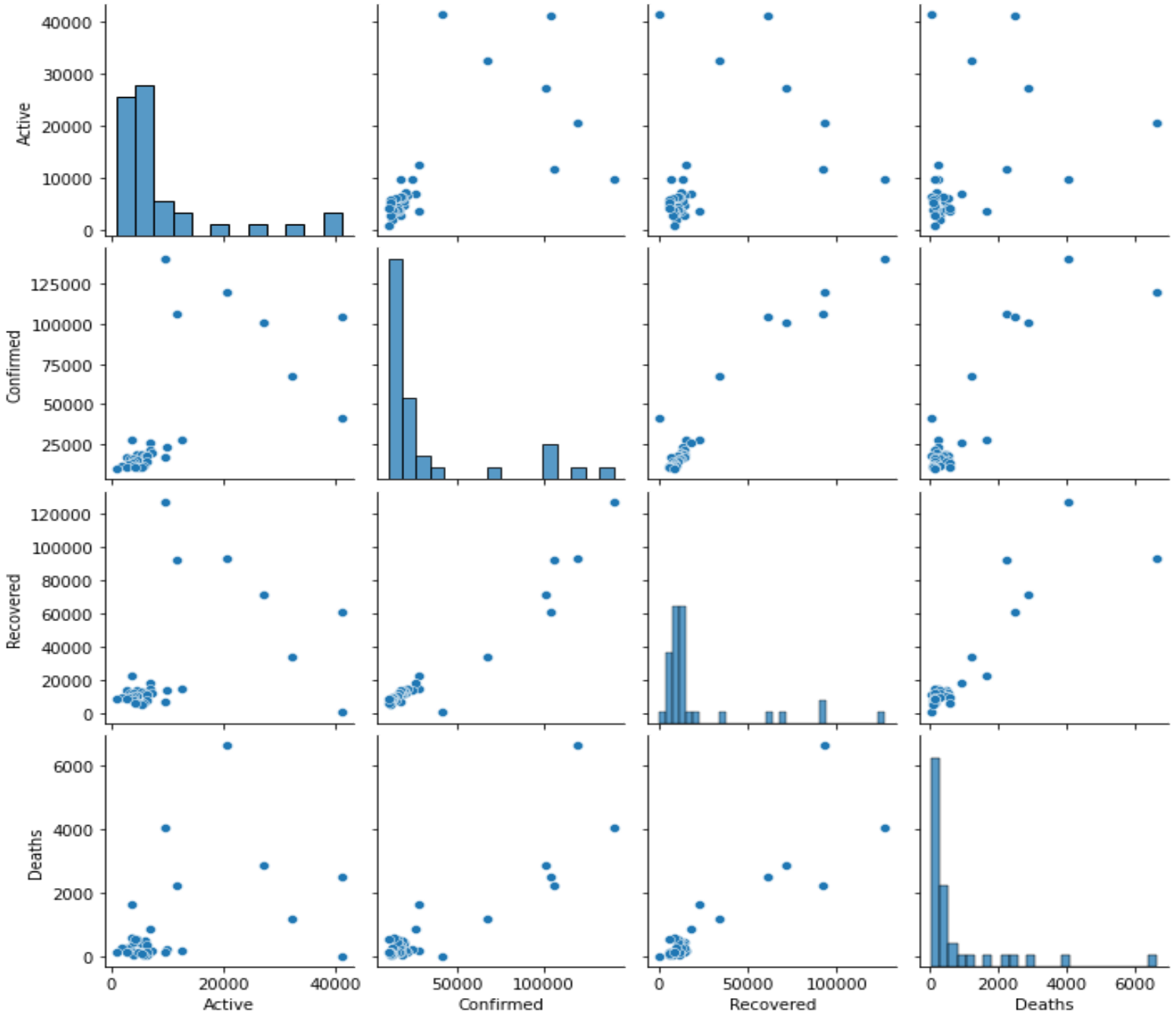
	Confirmed	Active	Recovered	Deaths	Migrated_Other	Delta_Confirmed	Delta_Active	Delta_Recovered	Delta_Deceased
Confirmed	1.000	0.661	0.946	0.900	0.417	0.688	0.554	0.665	0.744
Active	0.661	1.000	0.631	0.530	0.205	0.710	0.567	0.589	0.603
Recovered	0.946	0.631	1.000	0.900	0.412	0.542	0.437	0.565	0.650
Deaths	0.900	0.530	0.900	1.000	0.704	0.483	0.403	0.514	0.742
Migrated_Other	0.417	0.205	0.412	0.704	1.000	0.146	0.024	0.140	0.459
Delta_Confirmed	0.688	0.710	0.542	0.483	0.146	1.000	0.651	0.766	0.732
Delta_Active	0.554	0.567	0.437	0.403	0.024	0.651	1.000	0.775	0.691
Delta_Recovered	0.665	0.589	0.565	0.514	0.140	0.766	0.775	1.000	0.704
Delta_Deceased	0.744	0.603	0.650	0.742	0.459	0.732	0.691	0.704	1.000

Covid-19 Analysis

Pairplot :

Pairplot is used to show the distribution of each variable and the relationship between them. Here for the relation between each column with itself, we have represented it using the histogram while the other relationships we have shown it using scatter plot.

As seen in the above heatmap, the correlation between Confirmed and Recovered is highly positive and in the scatter plot, we can see that the values approximately lie in a straight line and for the value is scattering more the correlation between Active and Deaths.

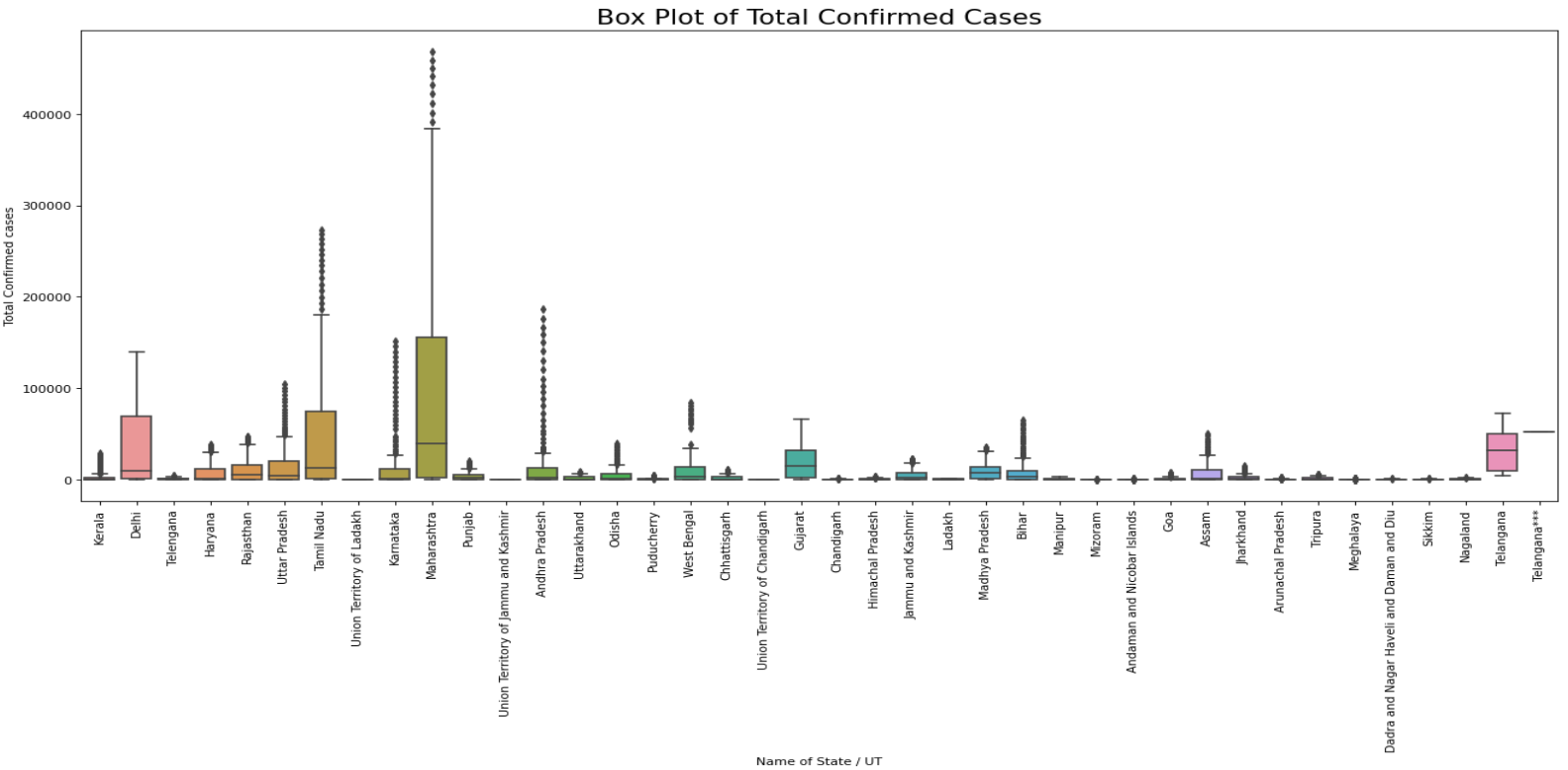


Covid-19 Analysis

Box Plot :

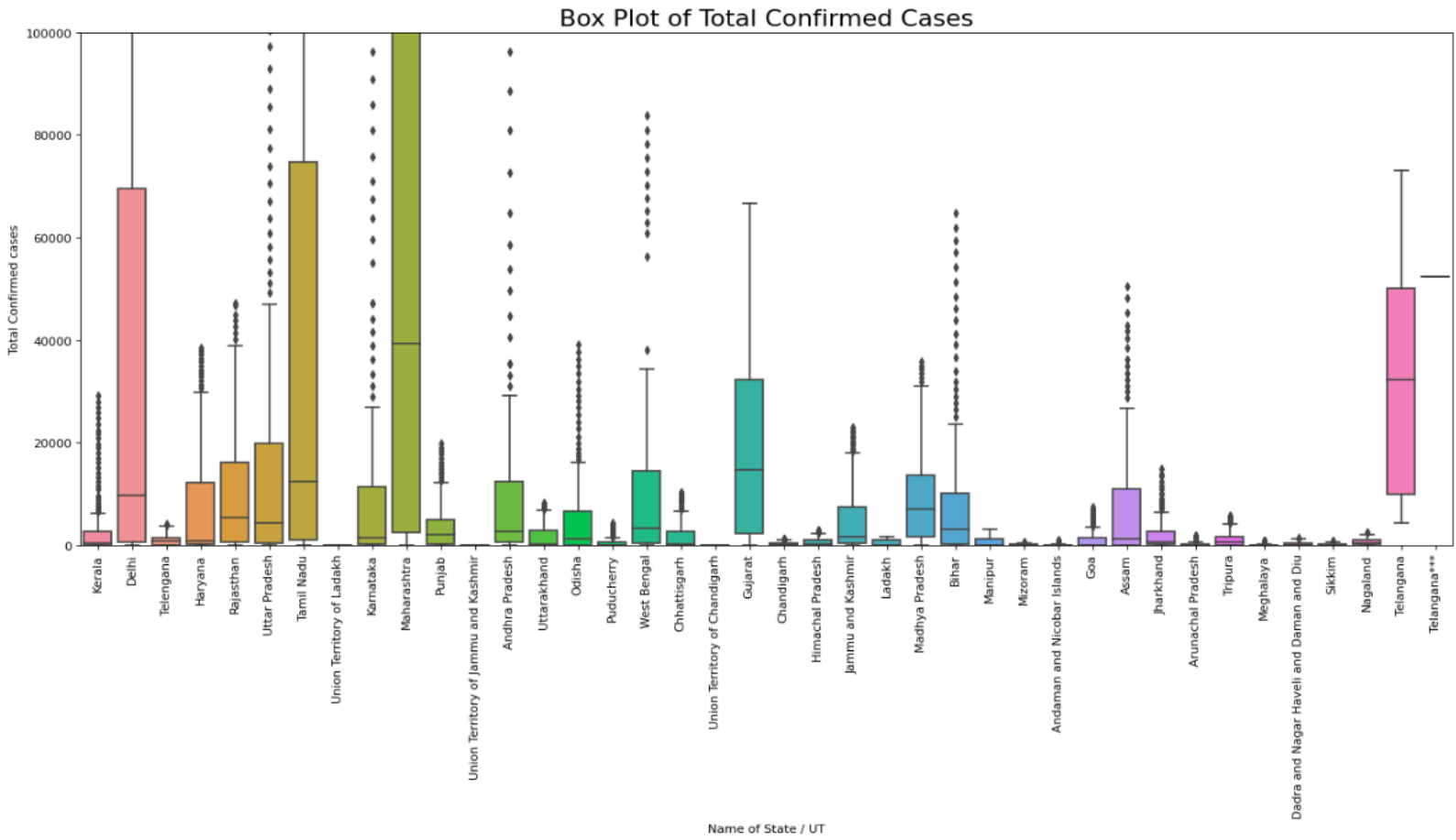
Box plot is used to describe the distribution across different variables. The first line indicates the minimum value; the last line indicates the maximum value, and the middle value indicates the median value. The box represents the range of the maximum value lying for the particular variable. And the dots seen are the outliers.

```
plt.figure(figsize=(20,8), dpi= 80)

sns.boxplot(x='Name of State / UT', y='Total Confirmed cases',
data=covid_data, notch=False)

plt.xticks(rotation=90)

plt.title('Box Plot of Total Confirmed Cases', fontsize=20)

plt.show()
```



Covid-19 Analysis

Taking the zoomed picture, we can see Telangana in the first line, which lies near 5500 is the minimum no. of confirmed cases, the second line, which lies near 70000, and the median line near 38000. The box in the middle ranges from about 12000 to 49000, indicating that the maximum number of cases lies between them.



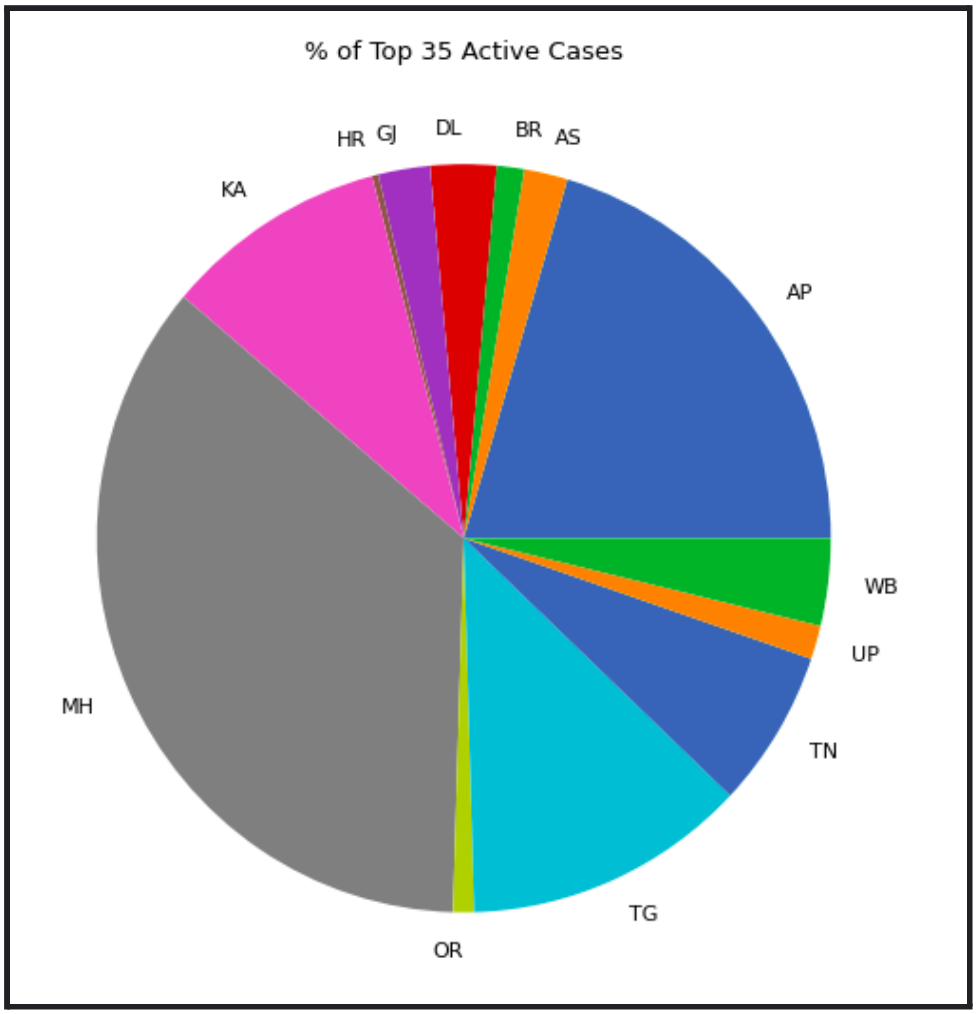
Pie Charts:

We have represented the pie charts to display the percentage share of cases present in our dataset.

So, we have displayed the percentage share of Top 35 active cases using the pie chart. Here we have grouped up the states. So what we are doing is summing up the top 35 cases according to their state and then storing its value in a variable called sums and then, with the help of the pie chart, representing the % share.

Covid-19 Analysis

```
plt.figure(figsize=(15,8), dpi= 80)

sums = top35.groupby(top35['State_Code'])['Active'].sum()

axis('equal');

pie(sums, labels=sums.index);

plt.title("% of Top 35 Active Cases")

show()
```



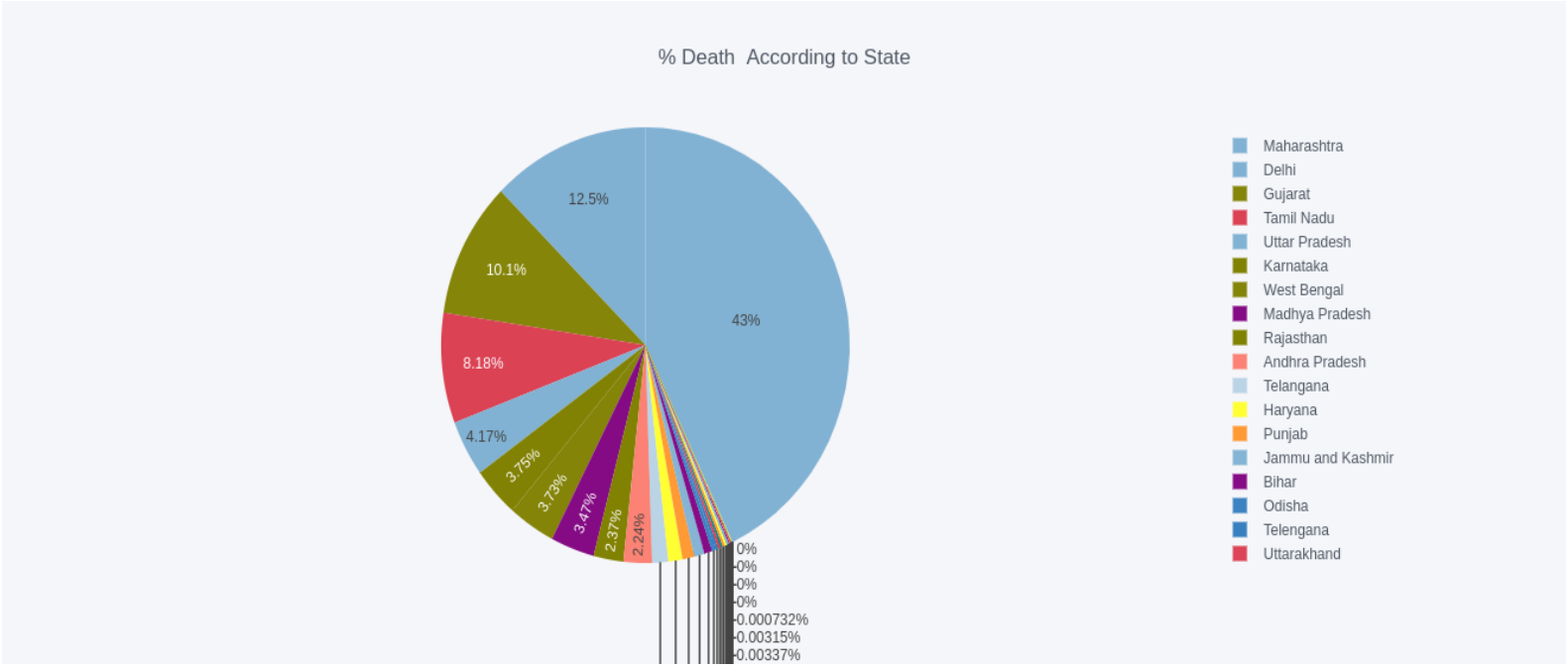
moving forward to our 2nd pie chart:

```
plt.figure(figsize=(20,20), dpi= 80)

data_plot=pd.DataFrame({'labels': covid_data['Name of State / UT'],'counts': covid_data['Death']})
```

Covid-19 Analysis

```
data_plot.iplot(kind='pie',labels='labels',values='counts',tit
le="% Death  According to State")

plt.show()
```

here we declare a variable called data_plot and store the percentage count of Deaths according to their states and then display the pie chart using **data_plot.iplot.**



As we can see, Maharashtra has the highest number of death cases at 43% followed by Delhi at 12.5%, Gujarat at 10.1%, and Tamil Nadu at 8.18%.

Line Graphs:

Line charts are used to represent the relation between two data X and Y on a different axis.

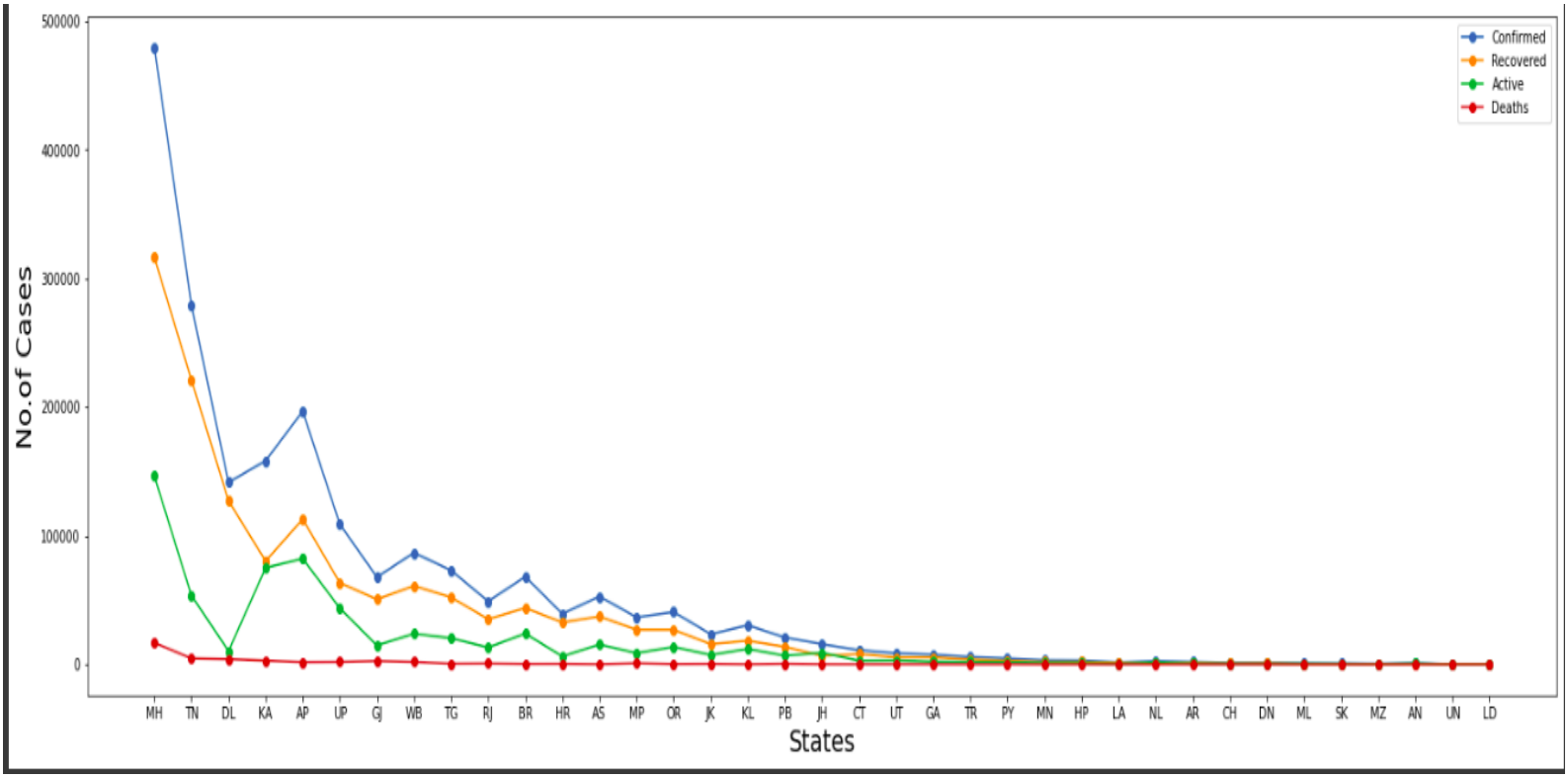
so, below we have shown the code of our first line graph, which shows the relationship between the no of cases and states.

Covid-19 Analysis

```
plt.figure(figsize=(25,8))
x = state_data ['State_code']
y = state_data ['Confirmed']
plt.plot(x,y,marker='o',label="Confirmed")
y = state_data ['Recovered']
plt.plot(x,y,marker='o',label="Recovered")
y = state_data ['Active']
plt.plot(x,y,marker='o',label="Active")
y = state_data ['Deaths']
plt.plot(x,y,marker='o',label="Deaths")
plt.xlabel('States',fontsize=20)
plt.ylabel('No.of Cases',fontsize=20)
plt.legend();
```

Now, here we are going to show the Number of confirmed cases, recovered cases, active cases, and deaths cases in one frame with respect to their states.

Covid-19 Analysis



As we can see that the blue line represents the number of Confirmed cases. Similarly, the orange line shows the no. of Recovered cases, the green line represents the number of Active cases and the red line represents the no. of Death cases.

If we just focus on the line that shows the no. of confirmed cases, we can see that the highest number of confirmed cases are present in Maharashtra state and are decreasing as we move towards Lakshadweep.

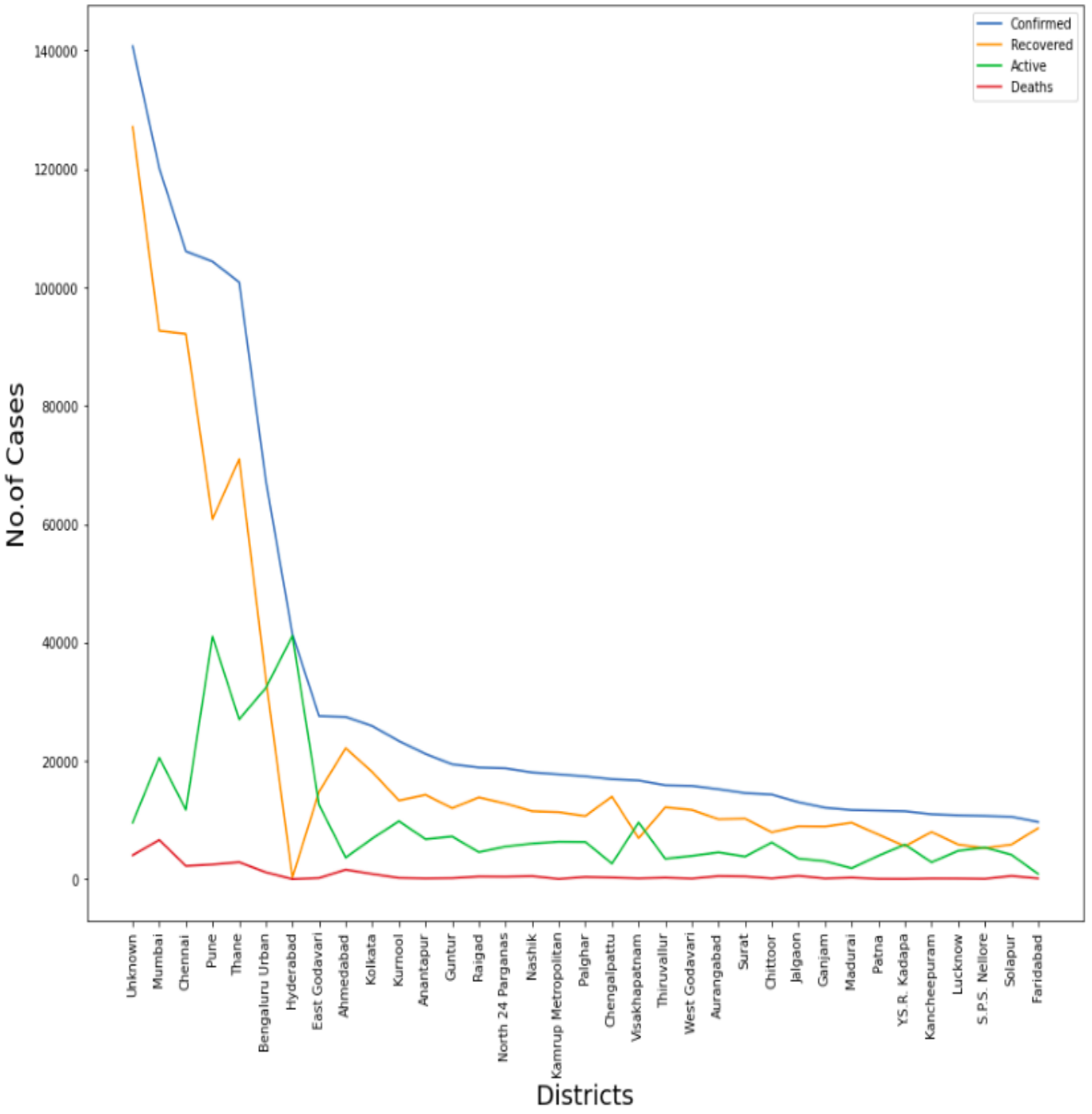
Moving forward to another line graph, we show the number of confirmed, recovered, active, and deaths cases in one frame with respect to their districts.

below is the code:

```
ind = np.arange(35)

width = 0.4


plt.figure(figsize=(15,12))

x = top35['District']

y = top35['Confirmed']

plt.plot(x,y,label="Confirmed")

y = top35['Recovered']

plt.plot(x,y,label="Recovered")

y = top35['Active']

plt.plot(x,y,label="Active")

y = top35['Deaths']

plt.plot(x,y,label="Deaths")

plt.xticks(rotation=90)

plt.xlabel('Districts',fontsize=20)

plt.ylabel('No.of Cases',fontsize=20)

plt.legend();
```

and now here we are displaying a plot of the line graph of the top 35 cases according to their districts.

Covid-19 Analysis



we can see that here also, the blue line represents the no. of Confirmed cases, orange represents the no. of Recovered cases, green represents no. of Active cases, red represents no. of Death cases.

As we can see in the above graph that there are different peak points for different types of cases, for example, if see the graph of no. of death cases, Mumbai has the highest number of death cases and similarly, if we see the graph of no. of active cases, Pune and Hyderabad are pretty close.

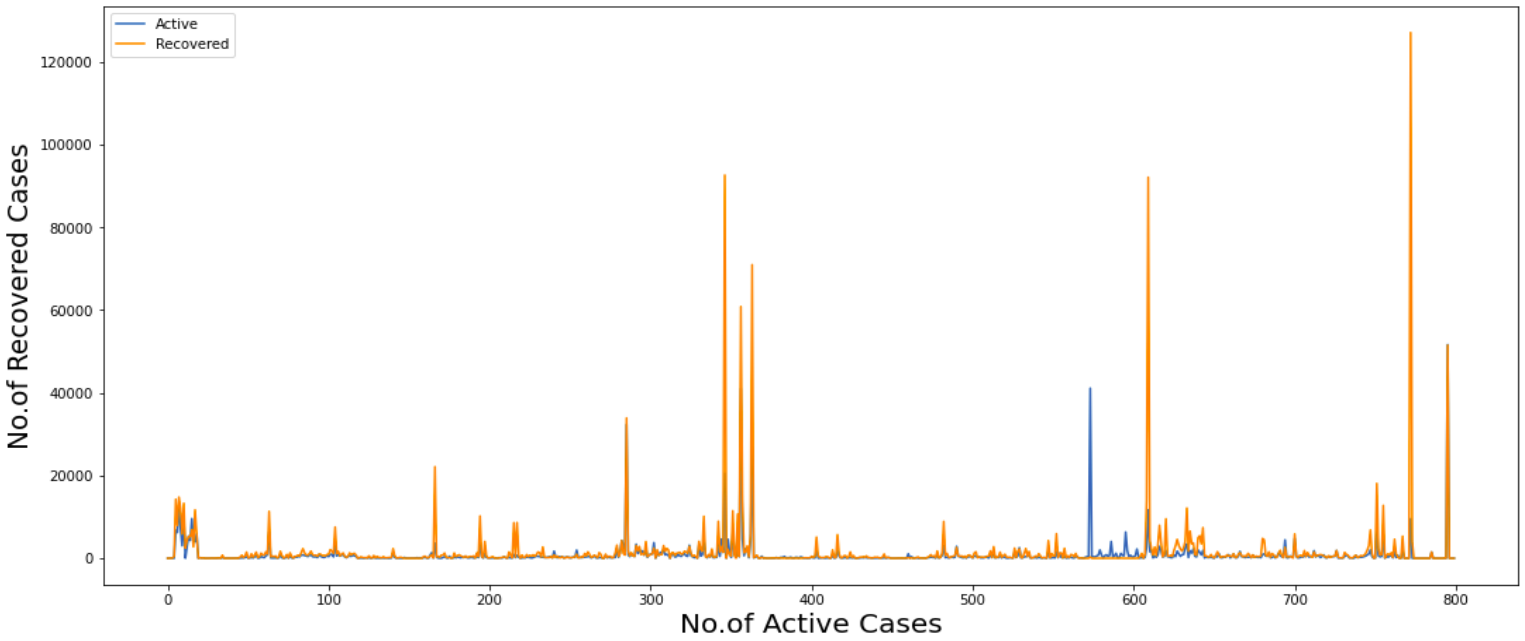
Covid-19 Analysis

Now Moving forward to our 3rd line graph that represents the no. of Active cases VS no. of Recovered cases.

below is the following code:

```
fig = plt.figure(figsize = (18,8))

x=district_data ['Active']

y=district_data ['Recovered']

plt.plot(x, label='Active')

plt.plot(y, label='Recovered')

plt.legend()

plt.xlabel('No.of Active Cases',fontsize=20)

plt.ylabel('No.of Recovered Cases',fontsize=20)

plt.show()
```

We are displaying the line graph showcasing the no. of active cases with respect to no. of recovered cases, the blue line shows the no. of active cases and the orange line shows the no. of recovered cases.



here the x-axis shows the no. of Active cases and the y-axis shows the no. of Recovered cases. we can see that the no. of Active cases, in the range of 550 to 600 are more than the no. of recovered cases. while other peak points show the no. of Recovered cases are more than the no. of Active cases.

Covid-19 Analysis

Conclusion

Through this program, we tried to perform a detailed data analysis on the Covid-19 outbreak in India. We have used different python libraries useful in performing Big-data analysis like NumPy, Pandas,Matplotlib,Plotly,Seaborn, etc. We performed various diagrammatic and statistical analyses on data generated from different sources using bar graphs, Scattered plots, pie charts, line graphs, Box plot and many more. Thus, providing an analytical view of the outbreak, which can further be used in planning and estimation.

Python provides a wide range of features for not just developing interfaces but also for handling & analyzing data. In the near future, we can even perform statistical tool of hypothesis testing and modelling using python libraries such as pandas, numpy, matplotlib etc, and can provide a conclusive remark to the analysis performed above in a computerized form.

Covid-19 Analysis

Bibliography

Link : https://pandas.pydata.org/

Link :https://matplotlib.org/

Link :https://numpy.org/

Link :https://plotly.com/

Link :https://seaborn.pydata.org/

Link :
https://www.kaggle.com/saurabhprakashgiri/covid-19-india-mar-aug-analysis-visualization/

Link :https://www.geeksforgeeks.org/