CAPSTONE PROJECT: BATTLE OF THE NEIGHBORHOODS
CORONA MAP

03/MAY/2020 - BY KARTHIK SREERAM

OVERVIEW

1. PURPOSE

This document provides the details of my final peer reviewed assignment for the IBM Data Science Professional Certificate program – Coursera Capstone.

2. INTRODUCTION

Coronavirus or COVID-19 needs no introduction. It has already been declared as a pandemic by WHO and in past couple of weeks it's impact has been deleterious from both health perspective and an economic one. Plenty has been written about it, especially statistical reports on its exponential growth and the importance of "flattening the curve.

Indian government has restricted people to move out of the house to reduce the transmission. People may go out of the house only to buy essentials such as medicine and groceries. So, people face lot of trouble due to the lockdown.

The intention on this project is to collect and provide a data driven approach about the corona spread and to find the location of pharmacy which is opened in the nearby location. So, people may avoid the red zone and buy essential medicine without worries.

I decided write a Python Script that pulls the latest Stagewise data of COVID-19 cases from the official website of Ministry of Health and Family Welfare, Government of India and turn it into insightful visualizations using popular Python packages like GeoPandas, Seaborn and Matplotlib and use foursquare api to get pharmacy shop details .

The python script will provide the following use case scenario:

1. State wise count of corona confirmed cases
2. Classify the states by different color according to the count
3. To know which medical shop (pharmacy) is opened in the particular city .
4. National wise count of recovered or died due to corona

3. DATA ACQUISITION

This demonstration will make use of the following data sources:

1. For Corona State Wise Count : Its Done By Using Web Scrapping On Official Indian GOVERNMNET CORONA SITE : https://www.mohfw.gov.in/
2. India Map Data : Downloaded From : https://map.igismap.com/share-map/export-layer/Indian_States/06409663226af2f3114485aa4e0a23b4
3. Pharmacy details data using Top Venue Recommendations from FourSquare API
   The following information are retrieved on the first query:

Venue ID
Venue Name
Coordinates : state and city

4. METHODOLOGY

Packages used

- Beautifulsoup — A library for pulling data out of *html* and *xml* files.
- Requests — A library for making HTTP requests in python.
- GeoPandas — A library for working with geospatial data in python.
- PrettyTable — quick and easy to represent tabular data in visually appealing ASCII tables.
- and other regular packages like Pandas, Matplotlib and Seaborn.

If you don't have any of the above mentioned packages installed on your system, please follow the installation instructions that are mentioned in the respective links.

(Note that Geopandas further depends on fiona for file access and Descartes and matplotlib for plotting)

Scrape the Data

To scrape a website using Python, you need to perform these four basic steps:

1. Sending an HTTP GET request to the URL of the webpage that you want to scrape, which will respond with the HTML content. We can do this by using the *Request* library of Python.
2. Analysing the HTML tags and their attributes, such as class, id, and other HTML tag attributes. Also, identifying your HTML tags where your content lives.
3. Fetching and parsing the data using *Beautifulsoup* library and maintain the data in some data structure such as Dictionary or List.
4. Output data in any file format such as csv, xlsx, json, etc. or use this tabulated data to make visualizations using *Seaborn/Matplotlib* libraries.

Extracting Data from HTML with Beautiful Soup

Formatting the data into machine readable . viewable format.

Beautiful soup is python packages which allows to extract data from html into beautiful python table

It is html parser for python .

Data clean up and pre-processing

Before we proceed further, notice that the scraped data columns are actually of ‘string’ datatype. We need to convert them into ‘int’ datatype.

Final ready data for processing:

Output:

Sr.No	States/UT	Confirmed	Recovered	Deceased
1	Andhra Pradesh	23	1	0
2	Andaman and Nicobar Islands	9	0	0
3	Bihar	15	0	1
4	Chandigarh	8	0	0
5	Chhattisgarh	7	0	0
6	Delhi	87	6	2
7	Goa	5	0	0
8	Gujarat	69	1	6
9	Haryana	36	18	0
10	Himachal Pradesh	3	0	1
11	Jammu and Kashmir	48	2	2
12	Karnataka	83	5	3
13	Kerala	202	19	1
14	Ladakh	13	3	0
15	Madhya Pradesh	47	0	3
16	Maharashtra	198	25	8
17	Manipur	1	0	0
18	Mizoram	1	0	0
19	Odisha	3	0	0
20	Puducherry	1	0	0
21	Punjab	38	1	1
22	Rajasthan	59	3	0
23	Tamil Nadu	67	4	1
24	Telengana	71	1	1
25	Uttarakhand	7	2	0
26	Uttar Pradesh	82	11	0
27	West Bengal	22	0	2
	Total	1205	102	32

exploratory data analysis

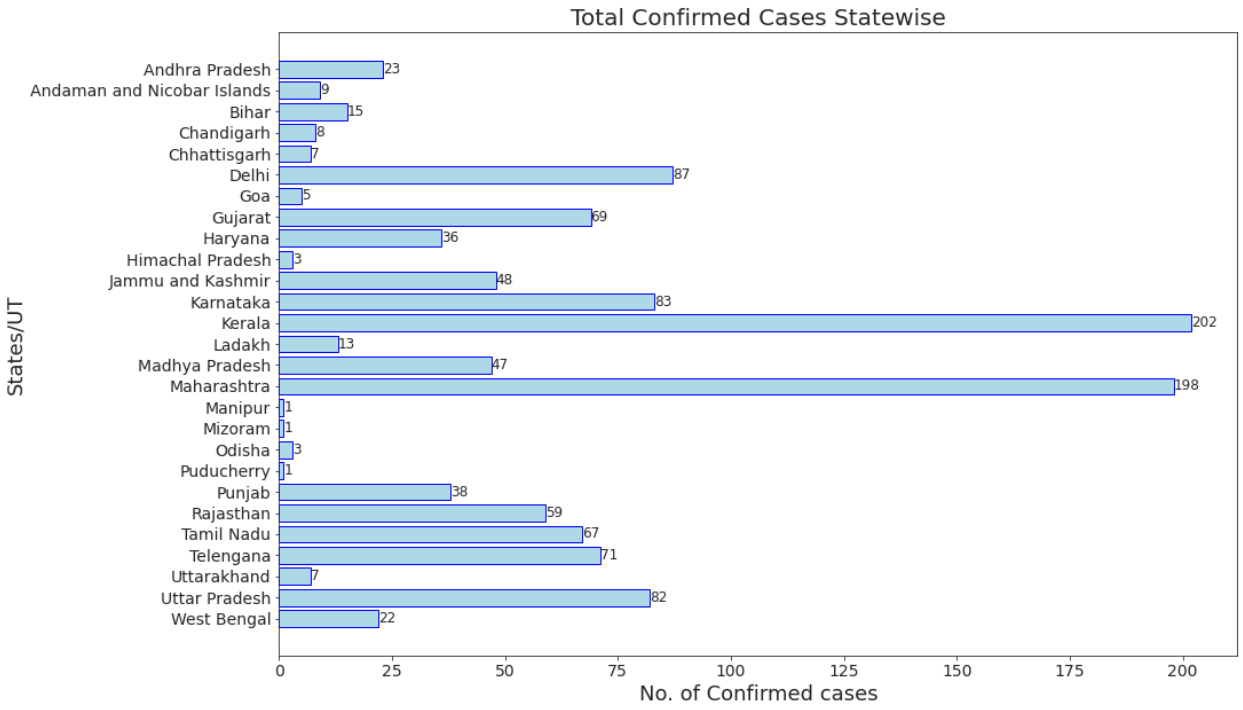
Reason for exploratory data analysis:

To make people understand how corona affected each state and also to categorize them based on the count

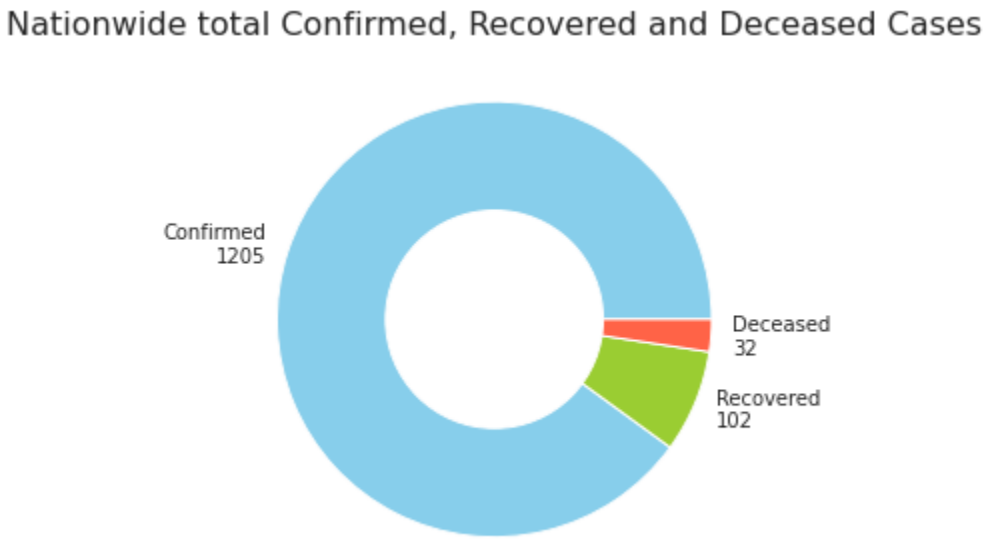
Bar Chart —State wise total confirmed cases

Plotting horizontal bar plot to show the state wise total confirmed cases

Output:



Donut Chart — Nationwide total Confirmed, Recovered and Deceased cases



5. Choropleth map of the total Confirmed Cases

The shape files used in this article to plot the India map with state boundaries can be downloaded from here.

Output:

	States/UT	geometry
0	Andaman & Nicobar Island	MULTIPOLYGON (((93.71976 7.20707, 93.71909 7.2...
1	Arunanchal Pradesh	POLYGON ((96.16261 29.38078, 96.16860 29.37432...
2	Assam	MULTIPOLYGON (((89.74323 26.30362, 89.74290 26...
3	Bihar	MULTIPOLYGON (((84.50720 24.26323, 84.50355 24...
4	Chandigarh	POLYGON ((76.84147 30.75996, 76.83599 30.73623...

I noticed that the names of some of the States and Union Territories(UT) and in the shape file were not consistent with the state names on the government website. So, I modified the State/UT names in the GeoDataFrame to match with the ones on our State dataframe.

Output:

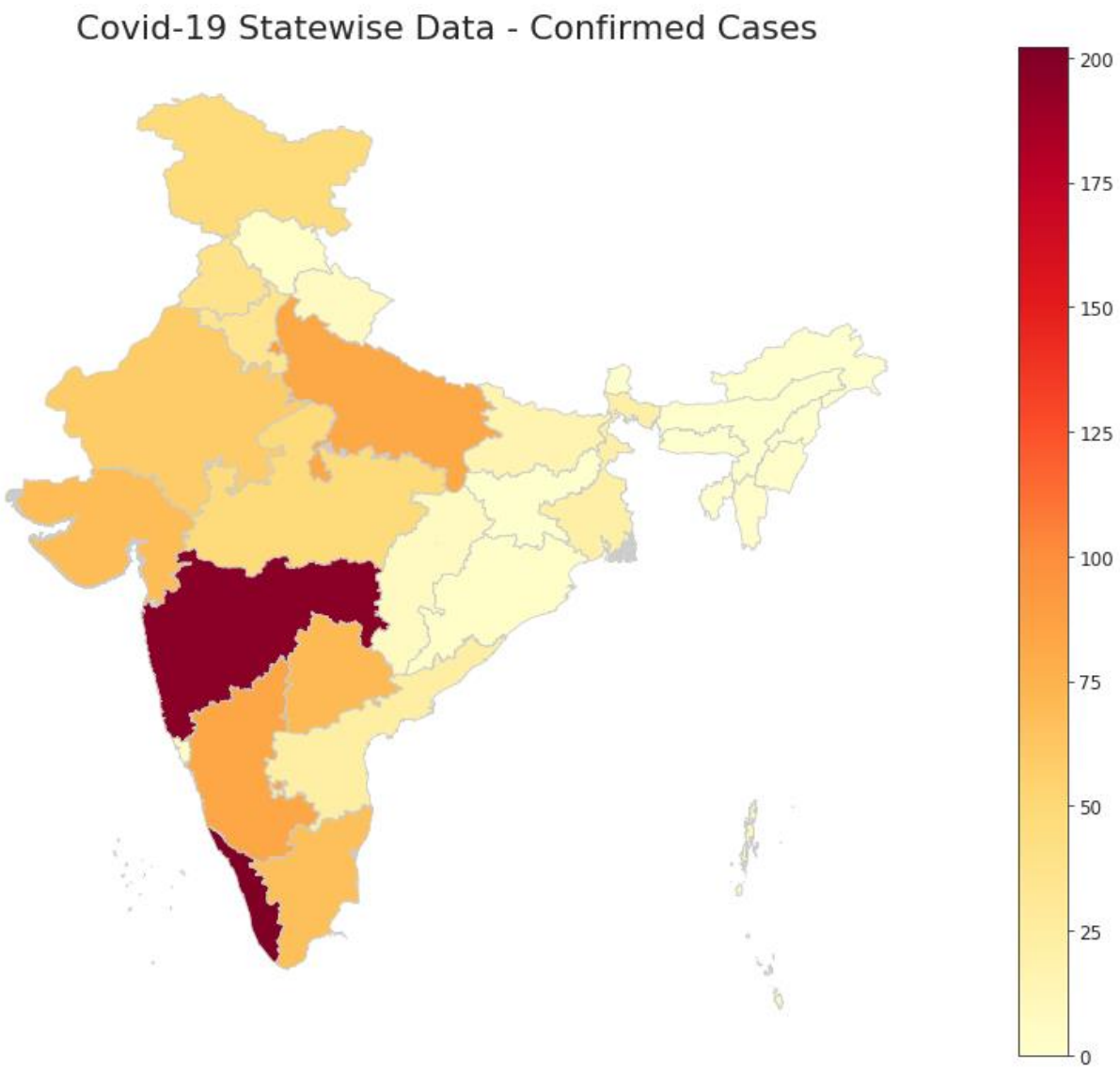
	States/UT	geometry	Confirmed	Recovered	Deceased
0	Andaman and Nicobar Islands	MULTIPOLYGON (((93.71976 7.20707, 93.71909 7.2...	9.0	0.0	0.0
1	Arunachal Pradesh	POLYGON ((96.16261 29.38078, 96.16860 29.37432...	0.0	0.0	0.0
2	Assam	MULTIPOLYGON (((89.74323 26.30362, 89.74290 26...	0.0	0.0	0.0
3	Bihar	MULTIPOLYGON (((84.50720 24.26323, 84.50355 24...	15.0	0.0	1.0
4	Chandigarh	POLYGON ((76.84147 30.75996, 76.83599 30.73623...	8.0	0.0	0.0

Then I merged the map data with government corona website scraped data to create colour code chart .

I did sta

Output:

Display the Statewise data on the map of India —



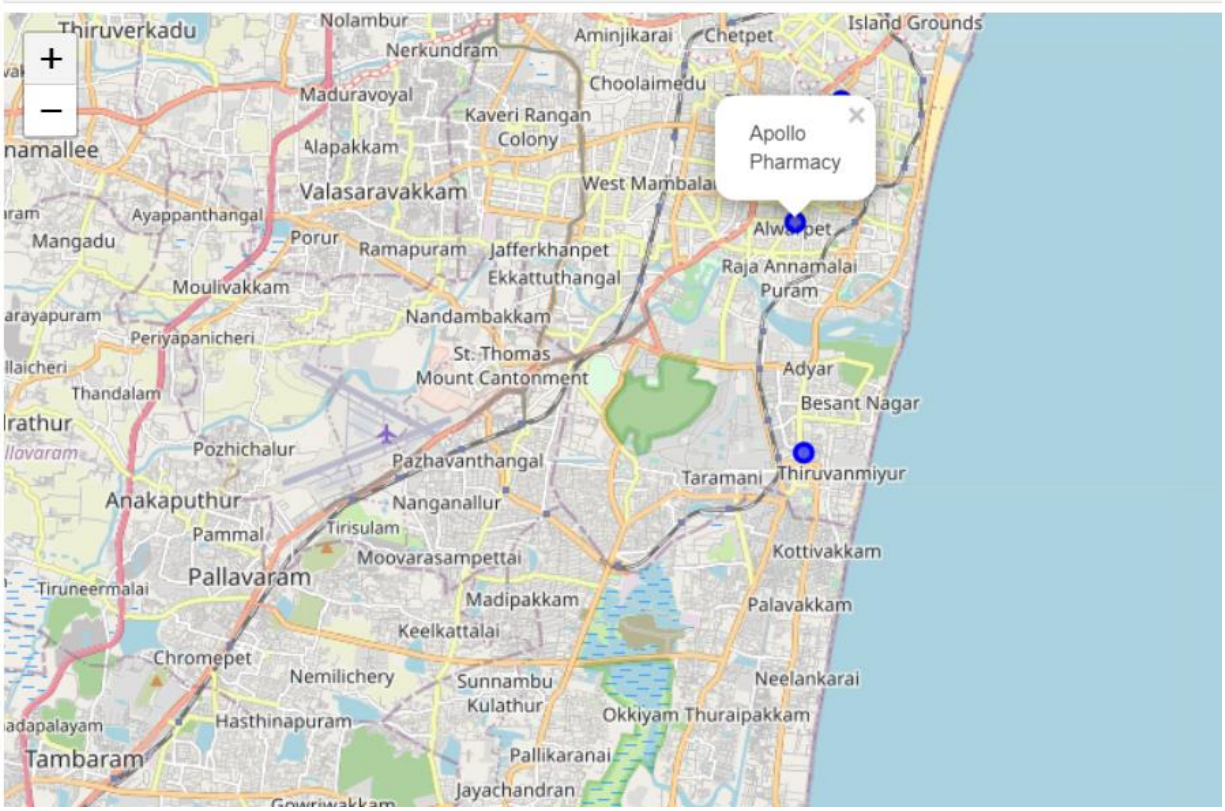
Magic with Foursquare:

Implementing Four square to find Pharmacy in the required given city.

Used four square api to fetch the pharmacy in the city required by the user .

Mapped the nearest pharmacy which is opened in the map , so user can go to the pharmacy without searching .

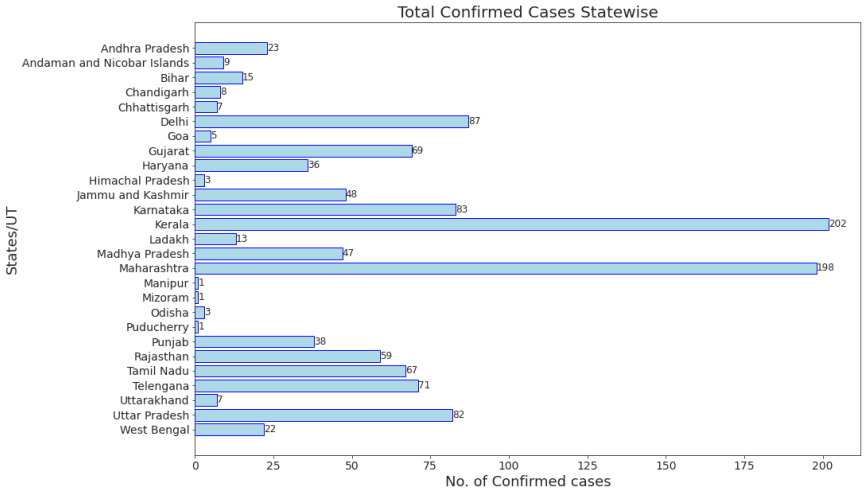
Output



5 Results section:

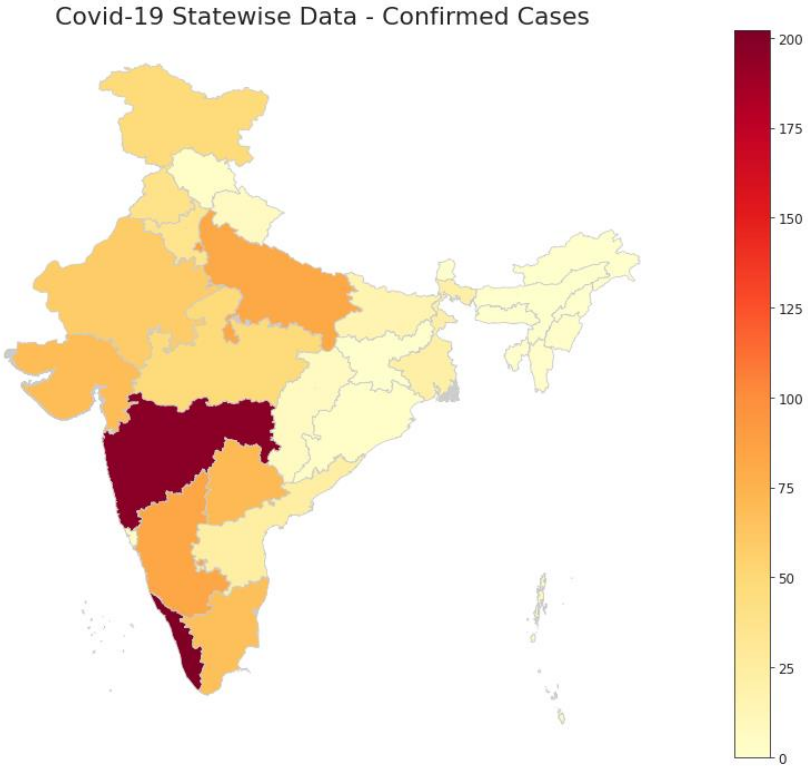
We have successfully implemented solution for all Business problem

1. State wise count of corona confirmed cases
    a. We have successfully showed the state wise count of corona



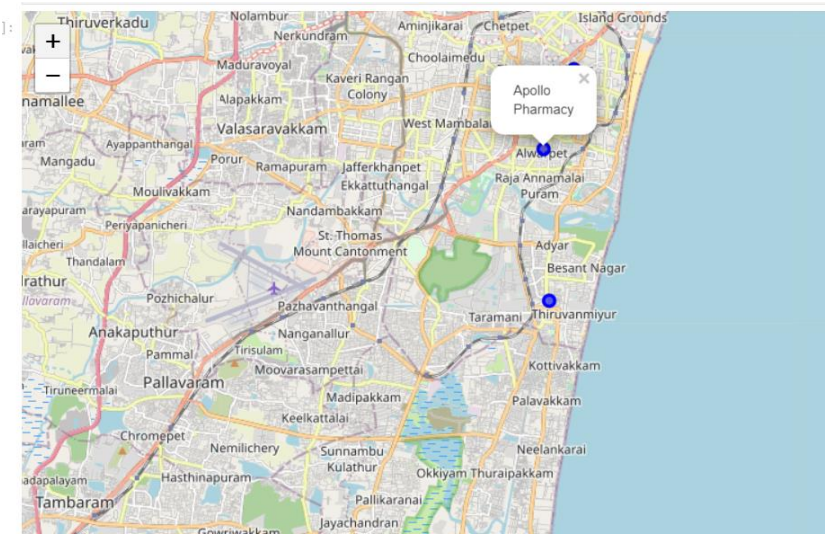
2. Classify the states by different color according to the count

   Yes we have successfully classified it



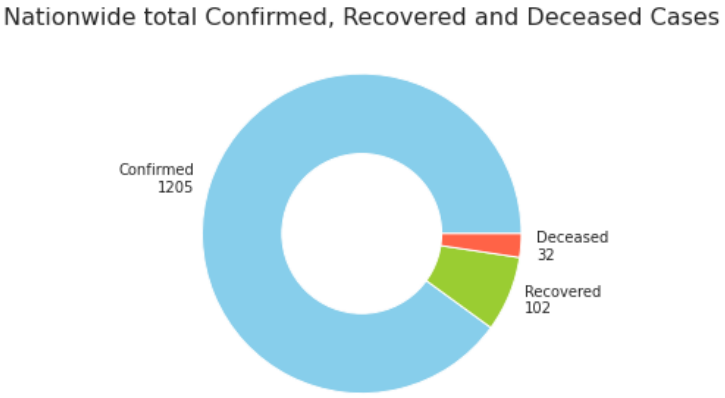
3. To know which medical shop (pharmacy) is opened in the particular city .

   Yes we used foresquare to identify the pharmacy in the city



4. National wise count of recovered or died due to corona.

   Yes we showed the national wise count of corona



6 Discussion section

From Classification of the states by different color according to the spread , we notice that Maharashtra has wide count . people in Maharashtra must be more careful and take extra preventive measure

Government must help in delivering essential things to people .

Detail code

```
import pandas as pd

import seaborn as sns

import matplotlib.pyplot as plt

import requests

from bs4 import BeautifulSoup

import geopandas as gpd

from prettytable import PrettyTable


# offical ministry of health website

url = 'https://www.mohfw.gov.in/'


# make a GET request to fetch the raw HTML content

web_content = requests.get(url).content
```

```
# parse the html content
soup = BeautifulSoup(web_content, "html.parser")

# remove any newlines and extra spaces from left and right
extract_contents = lambda row: [x.text.replace('\n', '') for x in row]

stats = [] # initialize stats
all_rows = soup.find_all('tr') # find all table rows

for row in all_rows:
    stat = extract_contents(row.find_all('td')) # find all data cells
    # notice that the data that we require is now a list of length 5
    if len(stat) == 5:
        stats.append(stat)

# now convert the data into a pandas dataframe for further processing
new_cols = ["Sr.No", "States/UT","Confirmed","Recovered","Deceased"]
state_data = pd.DataFrame(data = stats, columns = new_cols)

# converting the 'string' data to 'int'
state_data['Confirmed'] = state_data['Confirmed'].map(int)
state_data['Recovered'] = state_data['Recovered'].map(int)
state_data['Deceased']  = state_data['Deceased'].map(int)

# pretty table representation
table = PrettyTable()
table.field_names = (new_cols)
for i in stats:
    table.add_row(i)
```

```
table.add_row(["","Total",
               sum(state_data['Confirmed']),
               sum(state_data['Recovered']),
               sum(state_data['Deceased'])])
print(table)

# barplot to show total confirmed cases Statewise
sns.set_style("ticks")
plt.figure(figsize = (15,10))
plt.barh(state_data["States/UT"], state_data["Confirmed"].map(int),
        align = 'center', color = 'lightblue', edgecolor = 'blue')
plt.xlabel('No. of Confirmed cases', fontsize = 18)
plt.ylabel('States/UT', fontsize = 18)
plt.gca().invert_yaxis() # this is to maintain the order in which the states appear
plt.xticks(fontsize = 14)
plt.yticks(fontsize = 14)
plt.title('Total Confirmed Cases Statewise', fontsize = 20)

for index, value in enumerate(state_data["Confirmed"]):
    plt.text(value, index, str(value), fontsize = 12, verticalalignment = 'center')
plt.show()

# donut chart representing nationwide total confirmed, cured and deceased cases
group_size = [sum(state_data['Confirmed']),
              sum(state_data['Recovered']),
              sum(state_data['Deceased'])]

group_labels = ['Confirmed\n' + str(sum(state_data['Confirmed'])),
                'Recovered\n' + str(sum(state_data['Recovered'])),
                'Deceased\n'  + str(sum(state_data['Deceased']))]
```

```
custom_colors = ['skyblue','yellowgreen','tomato']


plt.figure(figsize = (5,5))
plt.pie(group_size, labels = group_labels, colors = custom_colors)
central_circle = plt.Circle((0,0), 0.5, color = 'white')
fig = plt.gcf()
fig.gca().add_artist(central_circle)
plt.rc('font', size = 12)
plt.title('Nationwide total Confirmed, Recovered and Deceased Cases', fontsize = 16)
plt.show()


# read the state wise shapefile of India in a GeoDataFrame and preview it
map_data = gpd.read_file('Indian_States.shp')
map_data.rename(columns = {'st_nm':'States/UT'}, inplace = True)
map_data.head()


# correct the name of states in the map dataframe
map_data['States/UT'] = map_data['States/UT'].str.replace('&', 'and')
map_data['States/UT'].replace('Arunanchal Pradesh', 'Arunachal Pradesh', inplace = True)
map_data['States/UT'].replace('Telangana', 'Telengana', inplace = True)
map_data['States/UT'].replace('NCT of Delhi', 'Delhi', inplace = True)


# merge both the dataframes - state_data and map_data
merged_data = pd.merge(map_data, state_data, how = 'left', on = 'States/UT')
merged_data.fillna(0, inplace = True)
merged_data.drop('Sr.No', axis = 1, inplace = True)
merged_data.head()


# create figure and axes for Matplotlib and set the title
fig, ax = plt.subplots(1, figsize=(20, 12))
```

```
ax.axis('off')

ax.set_title('Covid-19 Statewise Data - Confirmed Cases', fontdict = {'fontsize': '25', 'fontweight' : '3'})

# plot the figure

merged_data.plot(column = 'Confirmed', cmap='YlOrRd', linewidth=0.8, ax=ax, edgecolor='0.8', legend = True)

plt.show() code

citydb=pd.read_csv("in.csv")

citydb

#give city name  where we neeed to if any drug store (phramacy is opened)

cityname="Chennai"

requiredcitygeo= citydb.loc[citydb['city'] == cityname]

requiredcitygeo

clatitude=requiredcitygeo['lat']

clongitude=requiredcitygeo['lng']

CLIENT_ID = '*****'

CLIENT_SECRET = ********

radius=100000

VERSION=20180323

query='drugstore'


url =
'https://api.foursquare.com/v2/venues/explore?&client_id={}&client_secret={}&v={}&ll={},{}&radius={}&query={}'.format
(

    CLIENT_ID,

    CLIENT_SECRET,

    VERSION,

    clatitude,

    clongitude,

    radius,

    query)

#I dint use limit cos the business suition is during lockdown and so only few  medicalshops are opened in the required
city and I require all the shops.

resp = requests.get(url=url, params=params)
```

```
data = json.loads(resp.text)
print(data)
venues = data['response']['groups'][0]['items']
nearby_venues = json_normalize(venues)
# filter columns
filtered_columns = ['venue.name', 'venue.location.lat', 'venue.location.lng']
nearby_venues =nearby_venues.loc[:, filtered_columns]

# clean columns
nearby_venues.columns = [col.split(".")[-1] for col in nearby_venues.columns]
nearby_venues.head()
venues_map = folium.Map(location=[clatitude, clongitude], zoom_start=15)
for lat, lng, label in zip(nearby_venues.lat, nearby_venues.lng,nearby_venues.name):
    folium.CircleMarker(
        [lat, lng],
        radius=5,
        color='blue',
        popup=label,
        fill = True,
        fill_color='blue',
        fill_opacity=0.6
    ).add_to(venues_map)
```

7 Conclusion section.

On this notebook, Detail Analysis of corona cases in India Is been presented . We have successfully build a pharmacy recommending engine using foursquare.