Covid19 India tracker

Language – Python

Framework – flask

Deployment done using - heroku

Data sources :

Links-

https://www.mohfw.gov.in/

https://datahub.io/core/covid-19/r/countries-aggregated.csv

https://datahub.io/core/covid-19/r/worldwide-aggregated.csv

https://www.worldometers.info/coronavirus/

APIs-

https://api.covid19india.org/raw_data.json

csv files (from kaggle)

population_india_census2011.csv

StatewiseTestingDetails.csv

Code:

Css

```
body{
 background-color:"#000000";

}
```

<u>Python</u>

```
import numpy as np
import pandas as pd
import plotly.graph_objects as go
import dash
import dash_html_components as html
import dash_core_components as dcc
from dash import Dash
from dash.dependencies import Input, Output
from io import StringIO
import requests
from bs4 import BeautifulSoup
import plotly.offline as pyo
import os
from requests import request
import urllib.request
import json
from pandas.io.json import json_normalize
```

```
headers = {'User-Agent': 'Mozilla/5.0 (Windows NT 6.1; WOW64)
AppleWebKit/537.36 (KHTML, like Gecko) Chrome/56.0.2924.76
Safari/537.36'}


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]

# find all table rows and data cells within

stats = []

all_rows = soup.find_all('tr')

for row in all_rows:

    stat = extract_contents(row.find_all('td'))

# 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)
```

```
pop = pd.read_csv('population_india_census2011.csv')

pop.rename(columns={'State / Union Territory': 'States/UT'}, inplace =
True)


p = pd.read_csv('population_india_census2011.csv')

p.rename(columns={'State / Union Territory': 'States/UT'}, inplace = True)


list_ = state_data['States/UT'].unique()

df = p[p['States/UT'].isin(['list_']) == False]

state_data = pd.merge(state_data,df,on = 'States/UT')

state_data['Mortality rate'] =
(state_data['Deceased'].map(int)/state_data['Population'].map(int))*100


url="https://datahub.io/core/covid-19/r/countries-aggregated.csv"

s=requests.get(url, headers= headers).text


df_con=pd.read_csv(StringIO(s)) ## per country cases per day


url="https://datahub.io/core/covid-19/r/worldwide-aggregated.csv"


s=requests.get(url, headers= headers).text


df_world=pd.read_csv(StringIO(s)) ## worldwide-cases per day


d = pd.read_csv('AgeGroupDetails.csv')
```

```
d['Percentage'] = (d['Percentage'].str.strip('%').astype(float))


beds = pd.read_csv('HospitalBedsIndia.csv')

beds['Total'] =
beds['NumUrbanBeds_NHP18']+beds['NumRuralBeds_NHP18']+beds['Nu
mPublicBeds_HMIS']

beds=beds.iloc[:36]


tests = pd.read_csv("ICMRTestingLabs.csv")

cen = tests.groupby('state')['lab'].count().reset_index()


url="https://www.worldometers.info/coronavirus/"

s=requests.get(url, headers= headers).text

df5=pd.read_html(StringIO(s))

test5 = df5[0]['TotalTests']

country5=df5[0]['Country,Other']



def read_from_api(URL):

    response = request(url=URL, method='get')

    x = URL.split('/').pop(-1)

    x = x[:-5]

    elevations = response.json()

    rec = elevations[x]

    return json_normalize(rec)
```

```
df_raw_data =
read_from_api('https://api.covid19india.org/raw_data.json')

gender = df_raw_data.groupby('detectedstate')['gender'].count()

gdm = df_raw_data[df_raw_data['gender']=='M']

gen1 = gdm.groupby('detectedstate')['gender'].count().reset_index()


gdf = df_raw_data[df_raw_data['gender']=='F']

gen2 = gdf.groupby('detectedstate')['gender'].count().reset_index()


pop = pd.read_csv('population_india_census2011.csv')

pop.rename(columns={'State / Union Territory': 'States/UT'}, inplace =
True)


df_raw_data.rename(columns={'detectedstate': 'States/UT'}, inplace =
True)


df1 = pop[pop['States/UT'].isin(['list_']) == False]


df_raw_data = pd.merge(df_raw_data,df1,on = 'States/UT')


test = pd.read_csv('StatewiseTestingDetails.csv')


gender = df_raw_data.groupby('States/UT')['gender'].count()

gdm = df_raw_data[df_raw_data['gender']=='M']
```

```
gen1 = gdm.groupby('States/UT')['gender'].count().reset_index()


gdf = df_raw_data[df_raw_data['gender']=='F']

gen2 = gdf.groupby('States/UT')['gender'].count().reset_index()


gen1=pd.merge(gen1,df,on = 'States/UT')

gen1['gp']=(gen1['gender'].map(int)/gen1['Population'].map(int))*100


gen2=pd.merge(gen2,df,on = 'States/UT')

gen2['gp']=(gen2['gender'].map(int)/gen2['Population'].map(int))*100


i = test['Negative'].sum()

j = test['Positive'].sum()

test_s = {'label':['Negative test','Positive test'],'number':[i,j]}

data_p=pd.DataFrame(test_s)


t = {'Country,Other':country5,'number':test5}

p_=pd.DataFrame(t)

ac = p_[p_['Country,Other'] =='India']


#counting world data

a=df_world.shape[0]

Confirmed_world=df_world[['Date','Confirmed']].iloc[a-
1].reset_index().iloc[1,1]
```

```
Recovered_world=df_world[['Date','Recovered']].iloc[a-
1].reset_index().iloc[1,1]

Deaths_world=df_world[['Date','Deaths']].iloc[a-1].reset_index().iloc[1,1]


#adding 2 columns

state_data['Fatality rate'] =
(state_data['Deceased'].map(int)/state_data['Confirmed'].map(int))*100

state_data['Recovery rate']  =
(state_data['Recovered'].map(int)/state_data['Confirmed'].map(int))*100


# Plot Line Chart here


trace = go.Scatter(x=state_data['States/UT'], y=state_data['Confirmed'],

                mode='lines+markers',

                marker={'color': '#030808'}, name='Confirmed')


trace1 = go.Scatter(x=state_data['States/UT'], y=state_data['Deceased'],

                 mode='lines+markers',marker={'color':
'#DC143C'},name='Death')


trace2 = go.Scatter(x=state_data['States/UT'], y=state_data['Recovered'],

                 mode='lines+markers', marker={'color': '#00a65a'},
name='Recovered')


data = [trace, trace1, trace2]
```

```
layout = go.Layout(title='Confirmed vs Death vs Recovered in India',

                   xaxis={'title': '','automargin' : True},

                   yaxis={'title': 'Numbers'})


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




#line chart

trace3=go.Scatter(x=df_world['Date'],y=df_world['Confirmed'],mode='lines+markers',name='Confirmed')

trace4=go.Scatter(x=df_world['Date'],y=df_world['Deaths'],mode='lines+markers',name='Deaths')

trace5=go.Scatter(x=df_world['Date'],y=df_world['Recovered'],mode='lines+markers',marker={'color':'#00a65a'},name='Recovered')

data1=[trace3,trace4,trace5]

layout1=go.Layout(title='Rise in Covid19 cases per day in the world',xaxis={'title':'Date'},yaxis={'title':'Total cases'})

fig1=go.Figure(data=data1,layout=layout1)


#piechart

trace6=go.Pie(labels=d['AgeGroup'],values=d['Percentage'],hole=.3,textposition='inside', textfont_size=14)


data2=[trace6]


layout2=go.Layout(title='Age probability to get affected by the virus')
```

```
fig2=go.Figure(data=data2,layout=layout2)

#stacked bar graph
trace7=go.Bar(x=gen1['States/UT'],y=gen1['gp'], name='Male',
            marker={'color':'#00a65a'})

trace8=go.Bar(x=gen2['States/UT'],y=gen2['gp'], name='Female',
            marker={'color':'#a6a65a'})

data3=[trace7,trace8]

layout3=go.Layout(title='Gender probability of getting affected in several
states',
              xaxis={'title':'','automargin': True},
              yaxis={'title':'Gender Probability'})

fig3=go.Figure(data=data3, layout=layout3)

#bubble plot
trace9=go.Scatter(x=beds['State/UT'],y=beds['Total'],mode='markers',
              marker={'size':beds['Sno']})

data4=[trace9]
```

```
layout4=go.Layout(title='Hospital beds present in each state to fight
Covid',

        xaxis={'title':''},

        yaxis={'title':'Total no. of beds'})


fig4=go.Figure(data=data4,layout=layout4)


#bar plot for labs

trace10 = go.Bar(x=cen['state'],y=cen['lab'])

data5=trace10

layout5 =go.Layout(title='Testing centres in different states',

          xaxis={'title':'','automargin': True},

        yaxis={'title':'Number','automargin': True})

fig5 = go.Figure(data=data5,layout=layout5)


#pie chart

trace11=go.Pie(labels=data_p['label'],values=data_p['number'],textposition
='inside', textfont_size=14)


data6=[trace11]


layout6=go.Layout(title='Covid19 test results')


fig6=go.Figure(data=data6,layout=layout6)
```

```
options1=[
   {'label':'Recovery rate', 'value':'Recovery rate'},
   {'label':'Fatality rate', 'value':'Fatality rate'},
    {'label':'Mortality rate', 'value':'Mortality rate'}

]



options=[
   {'label':'Confirmed', 'value':'Confirmed'},
   {'label':'Recovered', 'value':'Recovered'},
   {'label':'Deaths', 'value':'Deaths'},
   {'label':'Total Tests for Covid19 done so far', 'value':'TotalTests'},

]

external_stylesheets = [
   {
        'href':
'https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css',
        'rel': 'stylesheet',
        'integrity': 'sha384-
MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ
8ERdknLPMO',
        'crossorigin': 'anonymous',


    }

]




app1 = dash.Dash(__name__, external_stylesheets=external_stylesheets)


server=app1.server

app1.layout=html.Div([

    html.H1("Covid19 India Tracker",style={'color':'#fff','text-
align':'center'}),

    html.Div([

        html.Div([

            html.Div([

                html.Div([

                    html.H3("Confirmed cases in India", className='text-light'),

                    html.H4(state_data['Confirmed'].map(int).sum(),
className='text-light')

                ], className='card-body')

            ], className='card bg-danger m-auto')

        ], className='col-md-3'),

        html.Div([

            html.Div([

                html.Div([
                    html.H3("Recovered cases in India", className='text-light'),
                    html.H4(state_data['Recovered'].map(int).sum(),
className='text-light')
                ], className='card-body')
            ], className='card bg-success m-auto')
        ], className='col-md-3'),
        html.Div([
            html.Div([
                html.Div([
                    html.H3("Death cases in India", className='text-light'),
                    html.H4(state_data['Deceased'].map(int).sum(),
className='text-light')
                ], className='card-body')
            ], className='card bg-warning h-100 m-auto')
        ], className='col-md-3'),
        html.Div([
            html.Div([
                html.Div([
                    html.H3("Active cases in India", className='text-light'),
                    html.H4((state_data['Confirmed'].map(int).sum()) -
(state_data['Deceased'].map(int).sum()) - (
                        state_data['Recovered'].map(int).sum()), className='text-
light')
                ], className='card-body')
            ], className='card bg-info h-100 m-auto')
        ], className='col-md-3')
    ], className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                    dcc.Dropdown(id='picker1',options=options
,value='Confirmed'),
                    dcc.Graph(id='choropleth')
                ],className='card-body')
            ],className='card bg-dark')
        ],className='col-md-12')
    ],className = 'row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                   dcc.Graph(id='line chart1',figure=fig1)
                ],className='card-body')
            ],className='card bg-dark')
        ],className='col-md-12')
    ],className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                    dcc.Graph(id = 'line chart',figure = fig)
                ],className='card-body')
            ],className='card bg-dark')
        ],className='col-md-12')
    ],className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                    dcc.Dropdown(id='picker', options=options1, value='Recovery
rate'),
                    dcc.Graph(id='bar')
                ],className='card-body')
            ],className='card bg-dark')
        ],className='col-md-12'),
    ], className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                    html.H3("Remember? Prevention is better than
cure",className='bold',style={'color': 'black', 'text-align': 'center'}),
                    html.H4("Check out your probability of getting affected and
stay safe", style={'color': 'black', 'text-align': 'center'})
                ], className='card-body')
            ], className='card bg-warning')
        ], className='col-md-12'),
    ], className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                    dcc.Graph(id='Pie1',figure=fig2)
                ], className='card-body')
            ], className='card bg-dark')
        ], className='col-md-6'),
        html.Div([
            html.Div([
                html.Div([
                    dcc.Graph(id='Bar',figure=fig3)
                ], className='card-body')
            ], className='card bg-dark ')
        ], className='col-md-6'),
    ], className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                    html.H3("Total number of Covid19 tests taken place in India
till date", className='text-light',style={'text-align': 'center'}),
                    html.H4(ac['number'].map(int), className='text-
light',style={'text-align': 'center'})
                ], className='card-body')
            ], className='card bg-info')
        ], className='col-md-12'),
    ], className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                   dcc.Graph(id='Pie chart',figure=fig6)
                ],className='card-body')
            ],className='card bg-dark ')
        ],className='col-md-6'),
        html.Div([
            html.Div([
                html.Div([
                    dcc.Graph(id='Bar1',figure=fig5)
                ], className='card-body')
            ], className='card bg-dark')
        ], className='col-md-6'),
    ], className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                    dcc.Graph(id='Bubble',figure=fig4)
                ], className='card-body')
            ], className='card bg-dark')
        ], className='col-md-12')
    ], className='row'),
    html.Div([
        html.Div([
            html.Div([
                html.Div([
                    html.H3("STAY HOME,STAY SAFE",
className='bold',style={'color': 'grey', 'text-align': 'center'}),
                    html.H6('@TEAM-SHIVAJI', style={'color': 'grey', 'text-
align': 'center'}),
                ], className='card-body')
            ], className='card bg-dark')
        ], className='col-md-12')
    ], className='row')


],className = 'container')
```

```
@app1.callback(Output('bar','figure'),[Input('picker','value')])

def update_graph(type):

   if type=='Recovery rate':

       return
{'data':[go.Bar(x=state_data['States/UT'],y=state_data['Recovery
rate'],marker_color='green')],

           'layout':go.Layout(title='Recovery rate in India',

              xaxis={'title':'','automargin' : True},

              yaxis={'title':'Recovery rate'})}

   elif type=='Fatality rate':

       return {'data':
[go.Bar(x=state_data['States/UT'],y=state_data['Fatality
rate'],marker_color='crimson')],

              'layout': go.Layout(title='Fatality rate in India',

              xaxis={'title':'','automargin': True},

              yaxis={'title':'Fatality rate'})}

   else:

       return {'data': [go.Bar(x=state_data['States/UT'],
y=state_data['Mortality rate'], marker_color='indianred')],

              'layout': go.Layout(title='Mortality rate in India',

              xaxis={'title': '', 'automargin': True},

              yaxis={'title': 'Mortality rate'})}
```

```
@app1.callback(Output('choropleth', 'figure'), [Input('picker1', 'value')])

def update_graph(type):

    if type == 'Confirmed':

         dff = df_con.groupby('Country')['Confirmed'].max().reset_index()

         return {'data': [go.Choropleth(locations=dff['Country'],
z=dff['Confirmed'],autocolorscale=False,

                                        locationmode='country
names',colorscale='rainbow',


marker={'line':{'color':'rgb(180,180,180)','width':0.5}},

                                        colorbar={'thickness':15,'len':1.,'x':0.9,'y':0.7,

                                        'title':{'text':'Confirmed','side':'bottom'}})],

                'layout': go.Layout(title='Confirmed cases all over the world, to
see where exactly India stands')}

    elif type == 'Recovered':

        dff1 = df_con.groupby('Country')['Recovered'].max().reset_index()

        return {'data': [go.Choropleth(locations=dff1['Country'],
z=dff1['Recovered'],autocolorscale=False,

                                        locationmode='country
names',colorscale='rainbow',


marker={'line':{'color':'rgb(255,255,255)','width':0.5}},

                                        colorbar={'thickness':15,'len':1,'x':0.9,'y':0.7,

                                        'title':{'text':'Recovered','side':'bottom'}})],

                 'layout': go.Layout(title='Recovered cases all over the world, to
see where exactly India stands')}

    elif type== 'Deaths' :
        dff2 = df_con.groupby('Country')['Deaths'].max().reset_index()

        return {'data': [go.Choropleth(locations=dff2['Country'],
z=dff2['Deaths'],autocolorscale=False,

                                    locationmode='country
names',colorscale='rainbow',


marker={'line':{'color':'rgb(255,255,255)','width':0.5}},

                                    colorbar={'thickness':15,'len':1,'x':0.9,'y':0.7,

                                    'title':{'text':'Deaths','side':'bottom'}})],

                'layout': go.Layout(title='Death cases all over the world,to see
where exactly India stands')}

    else:

        return {'data': [go.Choropleth(locations=country5, z=test5,
autocolorscale=False,

                                   locationmode='country names',
colorscale='rainbow',

                                   marker={'line': {'color': 'rgb(255,255,255)', 'width':
0.5}},

                                   colorbar={'thickness': 15, 'len': 1, 'x': 0.9, 'y': 0.7,

                                             'title': {'text': 'Total Tests', 'side': 'bottom'}})],

                'layout': go.Layout(title='Total Tests all over the world,to see
where exactly India stands')}




if __name__=="__main__":

    app1.run_server(debug=False)
```